ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Scienze
Corso di Laurea in Ingegneria e Scienze Informatiche

SIMILARITÀ SEMANTICA E CLUSTERING DI CONCETTI DELLA LETTERATURA MEDICA RAPPRESENTATI CON LANGUAGE MODEL E KNOWLEDGE GRAPH DI EVENTI

Elaborato in
Programmazione

Relatore
Prof. Antonella Carbonaro

Co-relatori
Prof. Gianluca Moro
Dott. Giacomo Frisoni

Presentata da
Giulio Carlassare

Terza Sessione di Laurea
Anno Accademico 2019 – 2020

PAROLE CHIAVE

Natural Language Processing

Event Extraction and Aggregation

Knowledge Graphs

Semantic Similarity

Rare Diseases

Gli uomini non hanno più il tempo per conoscere le cose.

Il Piccolo Principe

Sommario

Sul web è presente una grande quantità di informazioni principalmente in formato testuale e la diffusione dei social network ne ha incrementato la produzione. La mancanza di struttura rende difficile l'utilizzo della conoscenza contenuta, generalmente espressa da fatti rappresentabili come relazioni (due entità legate da un predicato) o eventi (in cui una parola esprime una semantica relativa anche a molte entità). La ricerca sta muovendo recentemente il proprio interesse verso i Knowledge Graph che permettono di codificare la conoscenza in un grafo dove i nodi rappresentano le entità e gli archi indicano le relazioni fra di esse. Nonostante al momento la loro costruzione richieda molto lavoro manuale, i recenti passi nel campo del Natural Language Understanding offrono strumenti sempre più sofisticati: in particolare, i language model basati su transformer sono la base di molte soluzioni per l'estrazione automatica di conoscenza dal testo. I temi trattati in questa tesi hanno applicazione diretta nell'ambito delle malattie rare: la scarsa disponibilità di informazioni ha portato alla nascita di comunità di pazienti sul web, in cui si scambiano pareri di indubbia rilevanza sulla propria esperienza. Catturare la "voce dei pazienti" può essere molto importante per far conoscere ai medici la visione che i diretti interessati hanno della malattia. Il caso di studio affrontato riguarda una specifica malattia rara, l'acalasia esofagea e il dataset di post pubblicati in un gruppo Facebook ad essa dedicato. Si propone una struttura modulare di riferimento, poi implementata con metodologie precedentemente analizzate. Viene infine presentata una soluzione in cui le interazioni in forma di eventi, estratte anche con l'utilizzo di un language model, vengono rappresentate efficacemente in uno spazio vettoriale che ne rispecchia il contenuto semantico dove è possibile effettuare clustering, calcolarne la similarità e di conseguenza aggregarli in un unico knowledge graph.

Introduzione

Sul web è presente una quantità incredibilmente vasta di informazioni e risorse, disponibili in molteplici forme. Tra tutte, il formato testuale è senza dubbio il più utilizzato, grazie anche alla sua semplicità e flessibilità. In particolare la sempre maggiore diffusione dei *social network* degli ultimi anni ha contribuito enormemente alla creazione di contenuti principalmente informali da parte degli utenti. La principale barriera che limita l'utilità di tutte queste informazioni è la loro mancanza di formalità e struttura, e la conseguente difficoltà nella ricerca, analisi e comprensione della conoscenza presente in esse. In particolare nel testo possono essere individuate le entità, cioè gli oggetti o concetti di cui si parla, le interazioni tra di esse e le relazioni che le legano. I **fatti** che emergono da queste interazioni possono essere rappresentati come relazioni che coinvolgono due entità collegate da un predicato (*"Mario"* → *"istituto di laurea"* → *"UniBo"*), oppure come eventi potenzialmente annidati che coinvolgono anche molte entità aventi un ruolo rispetto al *trigger* che indica la semantica (ad esempio il *trigger* è *"laureato"*, gli argomenti sono *"Mario"* con ruolo *laureando*, *"UniBo"* con ruolo *istituto* e *"110 e lode"* con ruolo *votazione*). Essi esprimono la conoscenza vera e propria espressa dal testo, e per poterla rappresentare in modo strutturato negli ultimi anni si è assistito a una crescita sempre maggiore dell'interesse verso i *knowledge graph*: permettono di rappresentare le conoscenza all'interno di una struttura a grafo, dove i nodi coincidono con entità di interesse e gli archi indicano le interazioni semantiche che sussistono fra di esse. I grafi sono strutture dati eleganti, che permettono di astrarre in modo diretto e intuitivo moltissimi scenari che si trovano nel mondo reale, offrendo una grande flessibilità; inoltre la teoria dei grafi si fonda su solide basi matematiche, utilizzabili per fornire una comprensione più profonda dei dati in essi contenuti. Progetti come *DBpedia*, *Wikidata* hanno lo scopo di racchiudere sempre più conoscenza in enormi *knowledge graph*; la loro costruzione richiede però una grande quantità di lavoro umano, a causa della complessità intrinseca nel linguaggio naturale (in particolare la sua ambiguità), della necessità di contestualizzare un'informazione per darle il giusto significato, e della grande quantità di errori (grammaticali, ortografici...) spesso presente nel testo informale proveniente dal web. Per questo, le informazioni che essi

contengono sono spesso limitate a un ambito molto generale, mentre sono tipicamente carenti in domini più circoscritti come una particolare disciplina scientifica o lo studio di una specifica malattia. Fortunatamente, i recenti passi avanti nell'ambito del *Natural Language Processing* (NLP) offrono strumenti e metodologie sempre più sofisticate: i risultati ottenuti da *language model* basati su architettura *transformer* (BERT, GPT-2...) hanno portato uno sconvolgimento nelle metodologie per l'analisi del testo. Tra le altre cose sono diventate la base di moltissime soluzioni ideate per l'estrazione automatica di conoscenza da testo in linguaggio naturale.

La presente tesi mira ad esplorare le tematiche legate all'utilizzo del *Natural Language Processing* per l'estrazione automatica di conoscenza dal testo e alla sua rappresentazione strutturata all'interno di *knowledge graph*. Tali tematiche rivestono un ruolo particolarmente significativo nell'ambito delle malattie rare: a causa della scarsa disponibilità di informazioni e della loro frammentazione, negli ultimi anni si è vista una forte crescita di comunità di pazienti sul web. I contesti social (quali i gruppi *Facebook*) divengono il territorio attraverso cui si scambiano informazioni di indubbia rilevanza durante tutto il percorso di un malato raro. Purtroppo però spesso rimangono all'interno di questo contesto e non vengono comunicate direttamente ai medici, rimanendo ignorate nel tempo: catturare la "voce dei pazienti" può essere un grande passo verso una maggiore inclusione e consapevolezza da parte di medici e ricercatori riguardo alla visione che i diretti interessati hanno della malattia.

Il caso di studio su cui verranno effettuati gli esperimenti riguarda una specifica malattia rara, l'**acalasia esofagea**. Il dataset utilizzato è composto da circa $67,000$ post pubblicati da pazienti e caregiver in un gruppo Facebook dedicato, gestito direttamente dalla *Associazione Malati Acalasia Esofagea* (AMAE)[1]. Verrà proposta una struttura modulare di riferimento e si analizzeranno diverse metodologie per la sua implementazione, estraendo e rappresentando in modo strutturato la conoscenza da testo in linguaggio naturale; per ogni implementazione saranno fatte alcune considerazioni sui risultati ottenuti.

Viene infine presentata una soluzione in cui le interazioni in forma di eventi, estratte anche con l'utilizzo di un *language model*, vengono rappresentate in uno spazio vettoriale dove è possibile effettuare *clustering*, calcolarne la similarità semantica e di conseguenza aggregarli in un unico *knowledge graph*. Data l'assenza di soluzioni di questo tipo in letteratura, vengono usati due modelli capaci di fornire individualmente soluzioni parziali, rispettivamente l'estrazione di conoscenza in forma di eventi e la rappresentazione di grafi in uno spazio vettoriale multidimensionale. Le due soluzioni insieme, modificate e adattate allo specifico scopo, sono in grado di rappresentare efficacemente la semantica

[1] `https://www.amae.it/`

degli eventi nello spazio vettoriale risultate.

Il documento è così organizzato:

- nel Capitolo 1 vengono presentate le tecnologie e le nozioni teoriche utilizzate nel resto della tesi, accompagnate da alcune metodologie interessanti disponibili in letteratura;
- nel Capitolo 2 si descrive il contesto di lavoro relativo alla acalasia esofagea, i dati utilizzati e si definiscono gli obiettivi;
- nel Capitolo 3 si individuano e analizzano nel dettaglio alcune soluzioni, delineando i pro e i contro per lo specifico caso di studio;
- nel Capitolo 4 si propone inizialmente una struttura modulare di riferimento con diverse implementazioni basate sulle soluzioni valutate nel capitolo precedente, per ognuna delle quali viene presentato il lavoro svolto e gli esperimenti effettuati con i relativi risultati;
- infine nel Capitolo 5 si traggono le conclusioni finali e si descrivono ulteriori possibili sviluppi.

Indice

Elenco delle figure

Capitolo 1

Framework teorico

In questo capitolo vengono presentate le tecnologie e le nozioni teoriche utilizzate nel resto della tesi.

1.1 Language Model basati su Transformer

L'architettura Transformer è stata proposta e descritta da Google Research [1], inizialmente con lo scopo della traduzione automatica. Lo sua struttura è illustrata in Figura 1.1.

L'obiettivo è quello di sostituire i modelli *sequence to sequence*[1] [2], usati per traduzione automatica, sintesi del testo, descrizione di immagini etc, che solitamente vengono implementati con reti neurali ricorrenti (RNN), in particolare *Long-Short Term Memory* (LSTM) o *Gated Recurring Unit* (GRU). La sequenza viene analizzata un elemento alla volta, iterativamente, da un componente chiamato *encoder*: ad ogni passaggio viene generata una nuova rappresentazione dell'input, combinando l'elemento corrente con la rappresentazione ottenuta dal passaggio precedente. Il risultato finale (che dovrebbe codificare il contenuto semantico dell'intero input) viene quindi fornito a un secondo componente chiamato *decoder*, che si occupa di costruire una nuova sequenza come output, ancora una volta in modo iterativo. Figura 1.2 illustra un esempio di questo funzionamento. Le RNN però soffrono del problema di *vanishing gradient*, un fenomeno per il quale i primi livelli della rete vengono addestrati molto più lentamente rispetto agli ultimi, richiedendo perciò tempi lunghi e molte risorse per ottenere buoni risultati. Inoltre a causa dell'elevato numero di passaggi intermedi tra la fase di codifica di un termine e la relativa decodifica, il modello non è in grado di catturare efficacemente le dipendenze tra

[1]Sono modelli usati in NLP che hanno sia come input che come output una sequenza di elementi (solitamente parole).

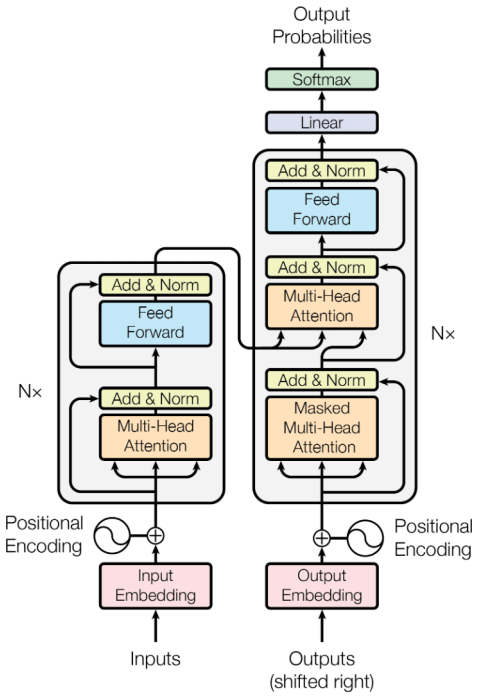


Figura 1.1: Architettura *Tranformer*
Fonte: *Attention is all you need* [1]

parole relativamente distanti nel testo, spesso presenti nel linguaggio naturale. Ad esempio nelle frasi *"The animal didn't cross the road because it was too tired"* e *"The animal didn't cross the road because it was too wide"*, il pronome *it* si riferisce rispettivamente a *The animal* e a *the road*; catturare il suo significato è indispensabile per poter ottenere una traduzione di qualità in lingue come l'italiano o il tedesco, assegnando il genere giusto all'aggettivo (*stanco* nel caso dell'animale, e *larga* nel caso della strada). Per risolvere questo problema nell'architettura *Transformer* viene rimossa la componente iterativa in favore del meccanismo dell'**attention**, che permette al modello di "concentrarsi" più su alcune parole che su altre, in modo da determinare il giusto contesto utile per dedurre il **significato** delle parole.

Il testo prima di essere processato viene convertito in *token* cioè singoli "termini" contenuti nel vocabolario del modello. In generale viene generato un *token* per ogni parola, ma in alcuni casi viene spezzata in pezzi più piccoli, soprattutto se è poco comune. Lo scopo di una granularità così fine è principalmente quello di contenere le dimensioni del vocabolario (che altrimenti dovrebbe considerare tutte le varianti di tutti termini conosciuti), ma permette anche al modello di gestire più correttamente parole composte: ad esempio *"effortless"* può essere suddivisa in *"effort"* e *"##less"* per separare il significato

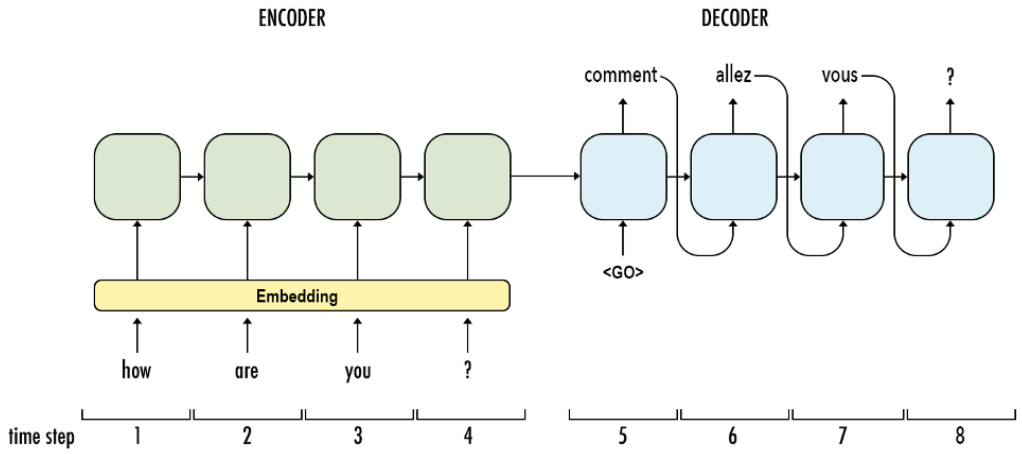


Figura 1.2: Schema di utilizzo di reti neurali ricorrenti per la traduzione automatica

Fonte: `https://towardsdatascience.com/language-translation-with-rnns-d84d43b40571`

primario dal modificatore. Il testo convertito in *token* viene poi analizzato.

Come visibile in Figura 1.1, un *Transformer* è formato da due blocchi principali: sulla sinistra è raffigurata la componente *encoder*, che nel caso della traduzione riceve come input il testo completo nella lingua originale. Per ogni *token* in ingresso viene calcolato un valore di *attention* verso **ogni altro token** dello stesso testo, sia quelli precedenti che successivi (*Self-Attention*). Lo scopo dell'*encoder* è quello di calcolare una rappresentazione che codifichi il contenuto semantico del testo in input.

Sulla destra della figura è presente invece il *decoder*: come input riceve un **prefisso di frase** (possibilmente vuoto) nella lingua obiettivo della traduzione e il suo scopo è quello di predire il prossimo *token* tradotto, utilizzando anche l'output dell'*encoder* che contiene il significato da tradurre. In questo caso, dato che l'input non è una frase completa, i valori di *attention* di ogni *token* fanno riferimento solamente a quelli precedenti (*Masked Self-Attention*, perché i valori verso i *token* successivi vengono rimossi).

In realtà sono presenti più componenti di *attention* chiamati *head* (solitamente 8 o 16) che operano in parallelo: per ogni *token* viene calcolato un valore verso gli altri per ogni *head* presente.

Questo approccio ha ottenuto risultati davvero strabilianti, e sulla base dell'architettura *Transformer* sono nati molti altri modelli con caratteristiche e obiettivi leggermente diversi; i principali sono BERT [3] (composto solamente dall'*encoder*), GPT [4] (composto solamente dal *decoder*) e T5 [5] (che invece usa

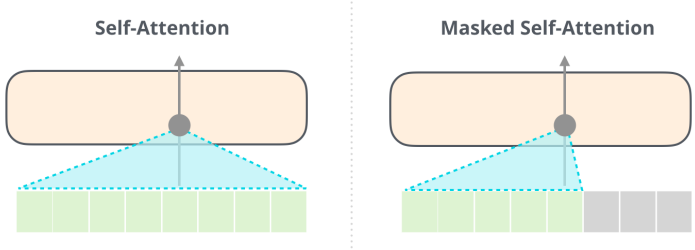


Figura 1.3: Nell'*encoder* ogni *token* ha un valore di *attention* verso ogni altro nel testo, mentre nel *decoder* solamente verso i *token* precedenti.
Fonte: `http://jalammar.github.io/illustrated-gpt2/`

il modello *Transformer* completo). Le componenti del *Transformer* utilizzate vengono sempre ripetute su più livelli in cui l'output di ognuno diventa l'input del successivo, per ottenere strutture più complesse.

In genere i modelli vengono prima addestrati su enormi dataset di dominio generico (ad esempio tutti gli articoli di Wikipedia), per poi fare *fine-tuning* per un compito specifico, cioè riaddestrarli parzialmente partendo dal risultato precedente.

I modelli *di base* citati sono stati poi realizzati in diverse versioni, variando le dimensioni (numero di parametri addestrabili), i dataset utilizzati e i valori degli iperparametri scelti in fase di addestramento. Di GPT esistono tre versioni (GPT, GPT-2, GPT-3), mentre di BERT sono state create moltissime varianti che hanno dato luogo alla cosiddetta BERTology [6]. In particolare le più interessanti per il nostro caso di studio (presentato nel Capitolo 2) sono senz'altro **SciBERT** [7] e **BioBERT** [8], addestrati rispettivamente su articoli scientifici ad ampio spettro, e su testo specifico dell'ambito biomedico.

1.2 Relation e Event Extraction

L'individuazione di interazioni semantiche tra entità menzionate nel testo è un obiettivo centrale del *Natural Language Processing* (NLP), dato che tramite esse viene espressa la conoscenza vera e propria.

Il modo più semplice per rappresentarle è tramite **relazioni**: si tratta di collegamenti tra coppie di entità, possibilmente diretti (non simmetrici) e con una specifica semantica (espressa da un predicato). Ad esempio la frase *"Mario si è laureato all'UniBo"* esprime una relazione tra *"Mario"* e *"UniBo"* con predicato *"istituto di laurea"*. Se le interazioni sono più complesse, soprattutto nell'ambito biomedico [9], vengono rappresentate come **eventi**, che si differenziano dalle relazioni perché possono connettere tra loro più di due entità, e possiedono una parola *trigger* presa dal testo (solitamente un verbo)

che descrive la semantica dell'interazione: le entità sono collegate al *trigger* come argomenti, a cui viene assegnato il ruolo che giocano rispetto ad esso. Inoltre gli eventi possono essere innestati, nel caso in cui l'evento esterno abbia un secondo evento come argomento, invece che un'entità. Ad esempio nella frase *"Mario si è laureato all'UniBo con 110 e lode"* il *trigger* è *"laureato"*, mentre gli argomenti sono *"Mario"* con ruolo *laureando*, *"UniBo"* con ruolo *istituto* e *"110 e lode"* con ruolo *votazione*. Gli eventi sono particolarmente adatti a rappresentare interazioni dinamiche (che descrivono un cambiamento o un'evoluzione, come l'esempio precedente), mentre per quelle statiche sono solitamente più adeguate le relazioni (ad esempio in *"Francesco è il padre il Giulio"*).

La maggior parte delle soluzioni per estrazione di eventi e relazioni parte dal presupposto che le entità siano state identificate precedentemente nel testo (vedi Sezione 1.3.2); suddividere i compiti però può introdurre un errore maggiore tra i due passaggi, anche perché il primo viene effettuato senza nessuna conoscenza del secondo e perciò non viene ottimizzato allo scopo. Per questo alcune soluzioni tentano di estrarre contemporaneamente entità e relazioni.

1.2.1 Event extraction

Le componenti essenziali in un evento sono il *trigger* (la radice dell'evento) e un insieme di argomenti, che possono essere entità o altri eventi; per ogni argomento è indicato il ruolo che ricopre nell'interazione. Spesso ad ogni evento è assegnata anche una tipologia che ne classifica la natura, in base al dominio di interesse.

Si parla di eventi a **dominio chiuso** quando le tipologie sono fissate e per ognuna è definita una struttura, che eventualmente descrive quali ruoli possono avere i suoi argomenti e con quali molteplicità (opzionali o obbligatori, multipli o singoli). Se non viene definito alcuno schema, si parla di eventi a **dominio aperto**. In alcuni casi viene definito uno schema fisso ma molto generico, basato sull'idea *"chi, ha fatto cosa, dove, quando, a chi, come"*.

Alcune soluzioni sono state proposte utilizzando reti neurali convoluzionali [10]: le parole del testo vengono prima trasformate in *embedding*, e sulla sequenza così ottenuta viene applicata la convoluzione, considerando piccoli gruppi di parole consecutive per ottenere un nuovo valore per ognuno. Tipicamente queste soluzioni risolvono il problema in due passaggi, inizialmente identificando il *trigger* e solo successivamente gli argomenti: in questo modo però non sono in grado di sfruttare appieno il contesto comune ad entrambi, e viene introdotto un certo errore tra i due passaggi (*pipeline*). Oltre a questo, la natura della convoluzione (considerare piccoli gruppi di parole consecutive) limita la capacità

di catturare e sfruttare interazioni distanti nel testo, e quindi le frasi nel loro complesso.

Di conseguenza nuove soluzioni sfruttano le reti neurali ricorrenti (RNN) [11, 12], in particolare con componenti *Long-Short Term Memory* (LSTM), che vengono utilizzate con successo a molti ambiti di NLP (vedi Sezione 1.1). In questo caso la sequenza di output contiene (per ogni termine) un valore che indica se si tratta di un *trigger* o di un argomento ed eventualmente di che tipo. Il problema quindi viene risolto in un unico passaggio, riducendo le problematiche citate in precedenza; le reti ricorrenti sono anche in grado di catturare dipendenze tra parole distanti, ma comunque con dei limiti (vedi Sezione 1.1). In [12] viene effettuata l'estrazione contemporanea di entità ed eventi.

Infine, come per altri compiti, le soluzioni basate su *transformer* hanno ottenuto i migliori risultati (vedi Sezione 1.1) con approcci simili a quelli utilizzati con le RNN. In [13] viene utilizzato SciBERT (Sezione 1.1) con tre ulteriori livelli neurali per effettuata l'estrazione di entità, *trigger* e argomenti in un unico passaggio.

1.2.2 Relation extraction

A causa della limitata disponibilità di dati etichettati, per poter fare addestramento supervisionato vengono talvolta creati dataset sintetici, generati automaticamente utilizzando euristiche e con l'aiuto di fonti di conoscenza esterna [14] (si parla di *distant supervision*). In altri casi vengono usate soluzioni non supervisionate, che solitamente sfruttano le capacità di modelli addestrati su altri compiti, ad esempio i *language model* [15, 16]. Nel primo caso il predicato viene scelto dall'insieme predeterminato usato durante l'addestramento [17]; nel secondo, non essendoci alcun dato etichettato, viene estratto direttamente dal testo, il che permette di avere una grande flessibilità e identificare molti tipi di relazioni, ma d'altro canto si ha una ridotta generalizzazione dato che la stessa semantica può essere espressa in modi diversi.

Anche in questo caso sono state proposte soluzioni con CNN [18] e LSTM [19], con problematiche simili a quelle riscontrate per l'estrazione di eventi. Per ridurre l'impatto di dati rumorosi (anche ad esempio in casi di *distant supervision*) è stato sperimentato l'uso del *reinforcement learning* [20], che ha il vantaggio di essere applicabile a quasi qualsiasi modello neurale. L'estrazione combinata di entità e relazioni ha ottenuto buoni risultati [21], anche se eguagliati da modelli in pipeline [22] (in cui i due passaggi vengono effettuati consecutivamente). L'utilizzo dei *language model* basati su *transformer* ha dimostrato ancora una volta la sua efficacia [17, 16, 23].

1.3 Knowledge Graph

Lo studio dei grafi ha visto una grande popolarità negli ultimi anni, grazie alle numerose applicazioni che trova nell'analisi delle reti sociali, della biologia computazionale e delle reti di calcolatori, per dirne alcuni; alla conferenza Neural Information Processing Systems (NeurIPS) 2020 ci sono stati 89 paper contenenti la parola *"graph"* nel titolo[2]. I grafi sono strutture dati eleganti, che permettono di astrarre in modo diretto e intuitivo moltissimi scenari che si trovano nel mondo reale, catturando in modo efficace le relazioni, potenzialmente cicliche, caratteristiche di reti sociali, interazioni biologiche, reti di trasporto, e così via. Allo stesso tempo hanno la capacità di rappresentare una struttura incompleta facendola evolvere nel tempo secondo le necessità. Inoltre la teoria dei grafi si fonda su solide basi matematiche, utilizzabili per fornire una comprensione più profonda dei dati in essi contenuti, che li rende uno strumento estremamente potente per una grande varietà di applicazioni.

Anno	**Graph**	**Tot**	%
2016	7	571	1.2
2017	10	648	1.4
2018	24	1018	2.3
2019	59	1434	4.1
2020	88	1912	4.6

Tabella 1.1: Numero di paper con la parola "Graph" nel titolo in vari anni di NeurIPS

In particolare i *Knowledge Graph* permettono di ottenere tutti i vantaggi sopra descritti, applicati alla rappresentazione della conoscenza. Il significato di *conoscenza* è in effetti molto vago, e tema di ampio dibattito in filosofia e gnoseologia[3]; quella rappresentata in un *knowledge graph* è **conoscenza fattuale**, che riguarda appunto i fatti (o proposizioni) che sono presenti nel mondo (o in un dominio di interesse). Esempio di conoscenza fattuale sono le frasi *"Bologna si trova in Emilia-Romagna"*, *"l'acqua è liquida"*, ma anche *"il mio nome è Giulio"*. Ogni fatto può essere suddiviso in tre componenti principali: un **soggetto**, un **oggetto** e un **predicato** che li lega; ad esempio *(Bologna → contenuta in → Emilia-Romagna)*, *(acqua → è un → liquido)*, *(io → ha nome → Giulio)*. È importante notare che in genere i predicati sono **unidirezionali**, cioè non è possibile scambiare soggetto e oggetto senza modificare la semantica: *(Emilia-Romagna → contenuta in → Bologna)* è

[2] `https://proceedings.neurips.cc/paper/2020`

[3] Termine filosofico equivalente a "teoria della conoscenza".

chiaramente diverso dall'esempio precedente; in alcuni casi è però possibile invertire il verso della relazione modificando il predicato (ad esempio *contenuta in* diventa *contiene*). Ci sono comunque relazioni simmetriche per cui l'ordine è irrilevante, come *(Giulio* $\leftrightarrow$ *amico di* $\leftrightarrow$ *Giacomo)*. Un insieme di fatti può essere facilmente rappresentato come un **grafo**, in cui soggetto e oggetto (più in generale **entità**) diventano nodi, messi in relazione dal predicato che diventa arco; il grafo è ovviamente direzionato per mantenere il verso delle relazioni.

Nonostante il termine *knowledge graph* con il significato attuale sia stato utilizzato per la prima volta da Google nel 2012 [24], ancora non esiste una definizione universalmente riconosciuta [25]; nell'ambito di questa tesi, sulla base delle considerazioni sopra descritte, indichiamo la seguente: *una struttura a grafo che rappresenta conoscenza sul mondo reale, in cui i nodi corrispondono a entità di interesse mentre gli archi rappresentano le relazioni che le legano*. Come detto, le relazioni possiedono una propria **semantica** che definisce il ruolo di una entità rispetto all'altra. Oltre alla struttura basilare indicata, spesso i *knowledge graph* contengono ulteriori attributi relativi sia alle entità (come ad esempio la classe a cui appartengono), sia alle relazioni (ad esempio il periodo o momento storico in cui la relazione è, è stata o sarà valida). Se l'insieme di fatti rappresentati diventa grande, si viene a creare una rete fortemente interconnessa che unisce concetti anche molto lontani tra loro, e su cui è possibile effettuare aggregazioni e calcolare statistiche in modo diretto per ottenere nuove visioni della conoscenza.

Negli ultimi anni si è visto un legame di supporto reciproco sempre maggiore tra *knowledge graph* e *machine learning*: in primo luogo il *machine learning* (in particolare NLP) viene usato per costruire, migliorare e completare i *knowledge graph* (chiamiamo questo compito **"Knowledge Graph learning"**); in secondo luogo, la conoscenza così rappresentata viene sfruttata per migliorare i risultati ottenuti da altri metodi o modelli. Questo utilizzo dei *knowledge graph*, talvolta chiamato **knowledge injection** [26], gode di grande interesse soprattutto in epoca recentissima, applicato con successo nell'ambito del *language modeling* [27, 28, 29], *question answering* [30, 31] e dei sistemi di raccomandazioni [32, 33], nonché per applicazioni nel dominio biomedico che richiedono una conoscenza estremamente precisa e specifica [34, 35].

1.3.1 Differenze con ontologie

Anche in questo caso, il dibattito è ancora aperto in letteratura su come esattamente un *Knowledge Graph* differisca da una ontologia [25, 36, 37] dato che entrambi sono rappresentati come grafi, in cui le entità vengono collegate tra loro in base alla relazione che le lega: i principali standard per la costruzione di ontologie, *Resource Description Framework* (RDF), *Web Ontology Language*

(OWL), vengono usati con successo anche per la rappresentazione di *Knowledge Graph* [38].

La differenza tra le due soluzioni non sta quindi nella loro struttura o dimensione, ma piuttosto nei dati stessi che esse rappresentano: i *Knowledge Graph* contengono **fatti**, mentre le ontologie definiscono uno **schema**. Esse non si concentrano sui dati (i fatti), ma su una definizione dettagliata dei concetti appartenenti a un particolare dominio di interesse e di come interagiscono tra di loro, aggiungendo spesso annotazioni per indicare sinonimi, definizioni, commenti o altri aspetti. Si può dire quindi che una ontologia rappresenta uno **schema**, definendo **classi** di concetti, mentre un *knowledge graph* contiene i dati, cioè **istanze** dei concetti contenuti nell'ontologia. Nel primo caso infatti, si tratta generalmente di piccole collezioni di informazioni curate manualmente da esperti, spesso per un utilizzo mirato in uno specifico dominio. Al contrario, un *knowledge graph* può contenere anche miliardi di relazioni, affiancando nozioni di dominio ad altre più generali.

1.3.2 Knowledge graph learning

Date la dimensione e la granularità dei *Knowledge Graph*, il lavoro richiesto per essere costruiti e curati in modo esclusivamente manuale è immenso. Vari progetti utilizzano diversi approcci per risolvere questo problema: WIKIDATA[4] si basa principalmente sul contributo di una grandissima comunità di volontari, organizzati in gruppi in base alla loro conoscenza e ai loro interessi, e solo una parte delle informazioni viene inserita e aggiornata da sistemi automatici; DBPEDIA[5] ha un sistema quasi totalmente automatico che estrae dati semi-strutturati dalle *infobox* di Wikipedia[6], ma che richiede comunque una verifica e validazione manuale da parte di un team dedicato [39].

Le difficoltà nella completa automazione della costruzione di *knowledge graph* da dati testuali non strutturati sono imputabili alla complessità intrinseca nel linguaggio naturale (in modo particolare la sua ambiguità), alla necessità di contestualizzare un'informazione per darle il giusto significato, e alla grande quantità di errori (grammaticali, ortografici...) spesso presente nel testo informale proveniente dal web. Per questo la ricerca si è concentrata principalmente sulla soluzione dei maggiori sottoproblemi, per alleggerire il lavoro umano necessario e con il fine ultimo di essere usati assieme per ottenere un unico sistema automatico.

[4] `https://www.wikidata.org/wiki/Wikidata:Main_Page`

[5] `https://www.dbpedia.org/`

[6] Sono i riquadri contenenti i dati più importanti riassunti in modo tabellare

Knowledge Graph Completion

Spesso i *knowledge graph* contengono solo un piccolo sottoinsieme della conoscenza, sia a causa delle risorse richieste come indicato sopra, sia perché molti fatti non vengono espressi esplicitamente nel testo. Lo scopo del *Knowledge Graph Completion* (KGC) è quello di determinare nuovi fatti da inserire in un *knowledge graph* esistente, tipicamente senza utilizzare risorse esterne e basandosi solo su quelle nel grafo stesso.

Due tipici problemi di KGC sono predizione di entità e predizione di relazioni: nel primo caso vengono fissati soggetto e predicato e si vuole predire l'oggetto, mentre nel secondo è il predicato a dover essere determinato. Tipicamente si sfruttano caratteristiche strutturali comuni all'interno del grafo: ad esempio per trovare il soggetto nella tripla $\langle$*"Italia", "capitale", ?* $\rangle$ si possono considerare le caratteristiche di altre capitali rispetto al Paese a cui appartengono per arrivare a dare la risposta.

Il primo approccio esplorato è quello degli *embedding* in cui le entità e i predicati vengono rappresentati in uno spazio vettoriale a bassa dimensionalità. Molti metodi rappresentano le relazioni come **traslazioni nello spazio** ($soggetto + predicato \approx oggetto$) [40, 41, 42], mentre altri propongono soluzioni più sofisticate, ad esempio usando reti neurali convoluzionali per rappresentare le relazioni con funzioni complesse [43].

I metodi basati su *embedding* si sono dimostrati incapaci di catturare relazioni con più passaggi[7] all'interno del grafo. Le relazioni con più passaggi sono estremamente importanti per dedurre nuovi fatti: ad esempio sapendo che *"Sara è nata a Faenza"* e che *"Faenza si trova in Italia"* si può dedurre che *"Sara è di nazionalità italiana"*. Per questo sono nati metodi che invece di limitarsi a cercare di rappresentare le entità considerandone i vicini, lavorano su interi percorsi in modo da catturare la struttura del grafo; gli approcci inizialmente logici e statistici [44] sono stati poi superati da quelli neurali [45]. Ulteriori miglioramenti sono stati ottenuti con l'utilizzo del *Reinforcement Learning* [46, 47].

Tutti i metodi esposti sono in grado di risolvere il problema di KGC con una certo livello di confidenza, ma non possono conoscere la realtà: una città potrebbe avere tutte le caratteristiche di una capitale senza esserlo, ad esempio per motivi storici o politici. Per questo è comunque necessario che i fatti così generati vengano verificati da esseri umani prima di essere integrati nel *knowledge graph*, per evitare un degrado della qualità e dell'utilità del grafo stesso.

[7] Che percorrono più di un arco consecutivamente.

Entity Extraction

In questo caso lo scopo è quello di riconoscere e catalogare le entità contenute nel testo in linguaggio naturale, in modo che possano poi essere utilizzate all'interno di un *knowledge graph*.

Il primo passo è **identificarle nel testo** (*entity recognition* o *named entity recognition*, NER). È molto frequente che i nomi delle entità siano formati da **più parole** (*"Regno Unito"*, *"Santa Sofia"*, oltre che tutti i nomi di persona completi), per cui solitamente vengono identificate non come singole parole ma come porzioni di testo contigue chiamate **menzioni**. È possibile che diverse menzioni nel testo facciano riferimento alla stessa entità che viene nominata più volte. Esistono soluzioni di vario tipo, ad esempio basate su LSTM [48] o su *language model* con architettura *transformer* [49] (vedi Sezione 1.1).

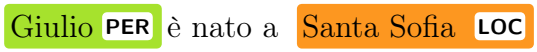


Figura 1.4: Un esempio di entità riconosciute con la loro classe ("persona" e "località").

A questo punto è possibile assegnare ad ogni entità una **classe** (ad esempio "PERSONA", "LUOGO", "DATA"...) per facilitarne l'uso in futuro (*entity typing*). Le classi possono essere più o meno specifiche e sono definite in base al dominio di interesse; spesso vengono organizzate gerarchicamente (in un albero), oppure all'interno di un'ontologia per una maggiore espressività. Le principali soluzioni utilizzano *embedding* per le etichette delle classi [50], talvolta ottenuti anche sulla base della struttura dell'ontologia che li contiene [51].

Infine è possibile collegare le menzioni identificate con le entità già contenute in un *knowledge graph* (*entity linking* o *entity disambiguation* o *named entity linking*, NEL), per integrare i fatti che le riguardano. Le menzioni vengono estratte direttamente dal testo e possono differire molto dal nome dell'entità a cui si riferiscono, ad esempio possono essere abbreviazioni (*"NBA"* che fa riferimento a *"National Basketball Association"*), incomplete (*"Einstein"* fa riferimento a *"Albert Einstein"*) oppure contenere degli errori (*"Benedit Cumberbatc"* fa riferimento a *"Benedi**c**t Cumberbatc**h**"*). Perciò è necessario tenere conto non solo del significato semantico della menzione in sé, ma anche del contesto in cui compare [51, 52].

Relation Extraction

Lo scopo è quello di estrarre le interazioni semantiche tra le entità descritte nel testo. Nella sezione 1.2 viene descritto questo compito nello specifico; è

importante notare che gli eventi non possono essere direttamente inseriti in un *knowledge graph* ma devono essere prima ristrutturati.

1.4 Clustering di grafi

Il **clustering di grafi** è una forma di *graph mining*: il *graph mining* è l'insieme degli strumenti e delle tecniche utilizzate per analizzare le caratteristiche dei grafi nel mondo reale, predire come la struttura e le proprietà di un dato grafo possano influire su alcune applicazioni di interesse, e sviluppare modelli che possano generare nuovi grafi che soddisfino particolari schemi trovati nel mondo reale [53].

Sono due le tipologie di *clustering* che possono essere effettuate su dati in forma di grafo [53, 54].

1. **Clustering inter-grafo**: lo scopo è suddividere un insieme (possibilmente molto grande) di grafi in gruppi in base al loro comportamento strutturale. I grafi vengono visti come oggetti che devono essere raggruppati, basandosi sulla similarità tra di essi. Ad esempio un insieme di grafi rappresentanti composti chimici vengono suddivisi in base alle loro caratteristiche strutturali.

2. **Clustering intra-grafo**: in questo caso si analizza un singolo grafo, e lo scopo è quello di suddividere i nodi (o gli archi): trovare gruppi di nodi "correlati" tra loro, secondo qualche criterio come la connettività dei vertici o la similarità dei vicini [55].

In letteratura, il termine *graph clustering* si riferisce solitamente alla seconda categoria, molto più popolare: in molti lavori viene considerata come l'unica interpretazione di *"graph clustering"*. Al contrario però, in questa tesi siamo interessati al **primo significato** (clustering di un insieme di grafi basandosi su somiglianze strutturali), che viene considerato **più complesso** dato che è necessario esaminare l'intera struttura dei grafi e il suo significato[8].

La maggior parte degli algoritmi classici di *clustering* tipicamente usa una funzione di distanza tra gli oggetti per misurarne la somiglianza; diventa quindi necessario definire una misura appropriata in grado di definire la distanza tra due oggetti strutturati. Inoltre, alcuni algoritmi (ad esempio K-Means) utilizzano oggetti rappresentativi (detti anche prototipi) come centroidi in alcuni importanti passaggi intermedi. Nonostante sia semplice ottenerli nel

[8]Il numero di sottografi cresce esponenzialmente all'aumentare della dimensione, per cui la loro enumerazione richiederebbe una grande quantità di memoria e tempo, rendendola insostenibile [56].

caso di oggetti multidimensionali, diventa molto più problematico quando si tratta di grafi. Di conseguenza devono essere progettati metodi appropriati per ottenere degli oggetti rappresentativi.

Gli approcci proposti in letteratura sono vari, ma spesso richiedono grafi con caratteristiche specifiche e non sono generalizzabili a situazioni diverse (presenza o mancanza di etichette sui nodi o sugli archi, variabilità delle dimensioni...). È inoltre comune trovare soluzioni che funzionano con grafi non direzionati, mentre in molte applicazioni la direzionalità degli archi rappresenta semantiche ben precise e non trascurabili (ad esempio nei *knowledge graph*).

- **Estrazione di caratteristiche** dai grafi, che poi vengono usate con metodi di clustering classici; si tratta di misure statistiche o topologiche, come il Coefficiente di Clustering, Indice di Assortatività, Average Shortest Path Length, Densità, Diametro, Centralità Media, etc.

- **Basati sulla topologia dei nodi**[9] [56], si fondano sull'assunzione che grafi simili contengano nodi con topologie simili; clustering inter- e intra-grafo vengono utilizzati iterativamente per ottenere risultati sempre più raffinati.

- **Multi-vista**, usano il classico clustering spettrale sulla somma pesata delle matrici di adiacenza normalizzate, *Singulare Value Decomposition*, *linked matrix factorization technique*, etc. Si assume che i grafi analizzati contengano informazioni riguardanti lo stesso insieme di oggetti (e quindi di nodi), cioè che siano **viste** differenti della stessa realtà.

- **Graph embedding** mira a trasferire i grafi in esame in uno spazio multidimensionale, permettendo poi di utilizzare gli algoritmi classici; il *graph embedding* è una funzione che mappa un grafo in un vettore n-dimensionale. Si trova a metà tra *Graph Analytics* (estrarre informazioni utili dai grafi) e *Representation Learning* (ottenere rappresentazioni di dati che rendano più semplice l'estrazione di informazioni utili): si occupa di generare rappresentazioni **a bassa dimensionalità** senza perdere informazioni sulla struttura e sulle proprietà del grafo. I *graph embedding* sono una tecnica efficace ed efficiente per sostituire metodi più classici, che spesso soffrono di un alto costo computazionale, combinando i vantaggi delle soluzioni statistiche e delle rappresentazioni strutturali.

[9] La topologia è una branca della geometria che studia le proprietà delle figure, e in generale degli oggetti matematici nello spazio, indipendentemente dalle distanze.

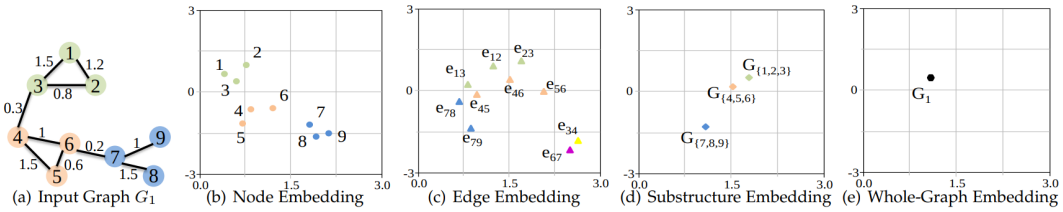


Figura 1.5: Un esempio di grafo con relativi *embedding* a due dimensioni. $G_{1,2,3}$ denota la sottostruttura contenente v_1, v_2 e v_3.
Fonte: [57]

1.4.1 Graph Embedding

Per le caratteristiche esposte, in questo lavoro si è scelto di utilizzare *graph embedding*; i metodi proposti sono solitamente generali, non limitati a una specifica categoria di grafi, rendendoli estremamente flessibili.

In particolare possiamo dividere i grafi in alcune categorie, sulla base delle loro caratteristiche.

- **Grafi omogenei**. Contengono solamente informazioni strutturali: né i nodi né gli archi possiedono attributi, e sono trattati tutti allo stesso modo. Le difficoltà stanno nel rappresentare in modo efficace gli schemi strutturali contenuti in essi.

- **Grafi eterogenei**. Al contrario contengono oggetti di tipologie diverse (sia nodi che archi) che devono essere rappresentati in uno spazio eterogeneo comune. Il problema principale è mantenere una coerenza tra il grafo di partenza e gli *embedding* generati; inoltre spesso la distribuzione delle diverse tipologie non è ben bilanciata. Un esempio sono i composti chimici dove, oltre alla struttura della molecola, ogni nodo corrisponde a un atomo di un particolare elemento e ogni arco a uno specifico tipo di legame.

- **Grafi con attributi**. Sono grafi eterogenei in cui i nodi, gli archi, o anche interi grafi possiedono specifici attributi aggiuntivi che descrivono caratteristiche dell'oggetto che viene rappresentato. I valori degli attributi devono essere presi in considerazione nella generazione degli *embedding*. I *knowledge graph* appartengono a questa categoria, dato che ogni nodo può appartenere a classi diverse (ad esempio "direttore", "attore", "film"...) e contenere anche un "nome" specifico ("Tim Burton", "Johnny Depp", "Edward mani di forbice") e gli archi possiedono un preciso significato ("produce", "dirige", "interpreta"...) oltre che una **direzione**.

Molto spesso l'*embedding* di un intero grafo viene calcolato in funzione degli *embedding* di alcune sue parti (nodi, archi, sottostrutture...); questi saranno chiaramente costruiti in modo da mantenere quante più informazioni possibili su eventuali attributi presenti in nodi e archi, e sulla struttura generale del grafo.

Deep learning

Nonostante siano stati proposti numerosi metodi analitici e algebrici [58, 59], l'interesse si sta spostando più verso approcci di *deep learning*, che hanno dimostrato grande efficacia in molteplici campi di ricerca come visione artificiale, *language modeling*... I modelli utilizzati per i grafi sono talvolta presi direttamente da altri ambiti e applicati come sono, oppure costruiti *ad-hoc*. L'input può consistere in una serie di **percorsi campionati** dal grafo in esame, oppure essere direttamente l'intero grafo.

Il primo caso (*Random Walk*) è quello utilizzato originariamente: il grafo viene rappresentato da una serie di cammini casuali all'interno di esso, e i modelli profondi vengono poi applicati sulle sequenze così ottenute. L'idea è che le caratteristiche strutturali del grafo vengano preservate all'interno dei percorsi, che essendo casuali dovrebbero coprire in modo uniforme nodi e archi. Un esempio è DeepWalk [60]: dal grafo viene inizialmente estratto un insieme di sequenze ottenute da cammini casuali, che vengono poi convertite in *embedding* usando SkipGram [61]. SkipGram è un modello progettato nell'ambito del NLP per generare *embedding* di parole contenute in un insieme di documenti, rappresentando il loro significato semantico sulla base del contesto in cui compaiono. In questo caso le "parole" sono i nodi del grafo, e i "documenti" i percorsi casuali; gli *embedding* vengono quindi calcolati sulla base del contesto dei nodi, in questo caso i vicini all'interno dei percorsi e di conseguenza all'interno del grafo. Una volta ottenuti gli *embedding* dei singoli nodi, è possibile usare altri modelli o tecniche già conosciute nel NLP (LSTM, GRU) per generare *embedding* di insiemi di parole (nodi) e ottenere quindi quello per il grafo completo.

La seconda classe di modelli riceve come input il grafo intero (solitamente nella forma di matrice di adiacenza); di seguito sono riportati gli approcci principali.

- **Autoencoder [62]** Una rete neurale viene addestrata a riprodurre la matrice di adiacenza (dato un nodo, vengono predetti i suoi vicini) e di conseguenza a catturare il "contesto" dei singoli nodi. Completato l'addestramento è possibile utilizzare lo stato interno della rete, a bassa dimensionalità, come *embedding*: ci si aspetta che nodi con vicini simili abbiamo *embedding* a loro volta simili.

- **Graph Neural Networks [63, 64] (GNN)** Sono una tipologia di reti neurali pensate specificatamente per lavorare su strutture a grafo, in particolare nel caso di grafi eterogenei o con attributi. Il principio su cui si basano è chiamato *message passing*: i valori associati ai nodi (in genere rappresentati come vettori) vengono "trasmessi" ai vicini e aggregati secondo una funzione non-lineare appresa dalla rete, eventualmente tenendo conto degli attributi degli archi; ripetendo questo procedimento più volte si ottiene un nuovo valore per ogni nodo in cui sono codificati i valori originali dei nodi, quelli degli archi e la struttura del grafo (che ha influenzato la propagazione dei "messaggi").

  Graph Convolutional Network (GCN) [65] sono un caso specifico di GNN che applica ai grafi la tecnica della convoluzione, utilizzata efficacemente in altri campi come l'analisi di immagini. Il funzionamento di base è lo stesso delle GNN, ma in questo caso dei filtri vengono applicati sul grafo per il riconoscimenti di determinati schemi (che vengono imparati dalla rete). Ad esempio SimGNN [66] utilizza delle GCN per generare *embedding* dei nodi, che poi vengono aggregati per ottenerne uno per l'intero grafo.

Dissimilarity Space Embeddings

Questo metodo è stato inizialmente ideato con lo scopo di mappare rappresentazioni strutturali di oggetti di interesse (non necessariamente grafi) in uno spazio vettoriale dove poter usare classici metodi statistici. In questo caso i grafi vengono rappresentati in un nuovo spazio vettoriale definito da vettori che misurano le dissimilarità degli gli oggetti di interesse X rispetto a un insieme R ($|R| = n$) di oggetti rappresentativi (prototipi); viene quindi usata la funzione di *embedding*

$$D(\cdot, R) : X \to \mathbb{R}^n$$

per ottenere uno spazio in cui è possibile usare il prodotto scalare e la distanza Euclidea [67]. La funzione di dissimilarità $d(x, r)$ può essere scelta in base all'applicazione desiderata, e gli *embedding* vengono calcolati come (vedi Figura 1.6)

$$D(x, R) \mapsto (d(x, r_1), d(x, r_2), ..., d(x, r_n))$$

Per calcolare la dissimilarità tra coppie di grafi esistono diverse metriche "algoritmiche" come *Graph Edit Distance* o *Maximum Common Subgraph*, ma sono tipicamente estremamente costose dal punto di vista computazionale e applicabili solamente a grafi molto piccoli (non più di 16 nodi [66]).

È possibile quindi utilizzare algoritmi approssimati [68, 69], oppure sfruttare l'uso delle **funzioni kernel**. Le funzioni kernel [70] permettono di confrontare

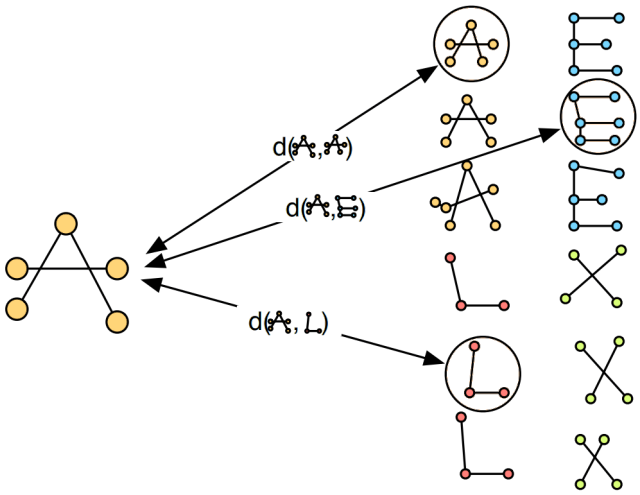


Figura 1.6: Ogni grafo viene confrontato con ognuno di quelli del set rappresentativo R.

Fonte: `https://www.informatik.uni-ulm.de/ni/ANNPR10/InvitedTalks/BuhnkeANNPR.pdf`

due oggetti appartenenti a uno spazio vettoriale con bassa dimensionalità, calcolando il prodotto scalare (ottenendo quindi una similarità) tra vettori in uno spazio con dimensionalità molto maggiore, senza la necessità di doverne calcolare le coordinate esplicitamente. Nonostante fossero inizialmente usate solamente con vettori numerici, è stata dimostrata [1] la loro efficacia anche quando applicati ai grafi.

Moderne soluzioni neurali permettono di calcolare *graph kernel* molto efficaci; SimGNN [66] sfrutta GCN per calcolare la similarità tra una coppia di grafi, mentre DDGK [71] utilizza un approccio non supervisionato per calcolare *graph kernel* e generare *embedding* nel *dissimilarity space*.

1.5 Similarità semantica del testo

Misurare la similarità semantica tra coppie di "contenuti testuali" (siano essi parole, frasi, o interi documenti) significa determinare quanto il significato dei due contenuti sia simile: gioca un ruolo importante in molti problemi di NLP come estrazione di informazioni, sintesi del testo, classificazione del testo, traduzioni automatica e *question answering*, per dirne alcuni. Purtroppo la versatilità del linguaggio naturale rende difficoltoso definire regole per misurare la similarità semantica. Non è sufficiente considerare la forma delle parole utilizzate, perché gli stessi termini possono avere significati diversi in base a come vengono usati (ad esempio le frasi *"Mario e Simone hanno studiato*

matematica e scienze" e *"Mario ha studiato matematica e Simone ha studiato scienze"* sono diverse anche se usano le stesse parole) oppure lo stesso concetto può essere espresso in modo differente (*"Maria è allergica ai prodotti caseari"* e *"Maria è intollerante al lattosio"* hanno significati estremamente simili). Un concetto simile è il *legame semantico* che misura invece quanto i due significati appartengono allo stesso contesto: ad esempio *"caffè"* e *"tazza"* vengono usati spesso assieme, ma il loro significato è differente. La similarità semantica può essere considerata come uno degli aspetti del legame semantico.

Le reti neurali profonde hanno superato i risultati ottenuti con metodi più classici, in particolare i *language model* basati su *transformer*; il loro utilizzo richiede però grandi risorse computazionali rispetto ad altre soluzioni. Inoltre vengono usate solitamente come "scatole nere" in cui è difficile capire gli elementi che hanno portato al risultato ottenuto.

Molte soluzioni utilizzano in primo luogo gli *embedding* delle parole: possono essere sfruttati modelli preaddestrati (ad esempio WORD2VEC [61, 72] e GLOVE [73]) per ottenere, data una parola, una sua rappresentazione vettoriale che ne cattura il significato. Per confrontare due documenti è possibile utilizzare direttamente gli *embedding* delle parole contenute [74] o combinarli per ottenere *embedding* dell'intero documento [75].

Alcuni metodi calcolano direttamente gli *embedding* dei documenti (o delle frasi) [76, 77]. Rappresentare i documenti in un unico spazio vettorial comune permette di confrontare un grande numero di coppie e di effettuare operazione come il *clustering*. In altri casi invece viene calcolata la similarità tra una singola coppia di documenti: il risultato è solitamente più accurato di quello ottenuto usando gli *embedding*, ma è necessario valutare singolarmente ogni coppia che può risultare estremamente costoso se i documenti sono molti. I *language model* basati su *transformer* hanno ottenuto ottimi risultati in questo compito: ROBERTA [78] addestrato sul *Semantic Textual Similarity Benchmark* [79] detiene attualmente lo stato dell'arte.

Capitolo 2

Caso di studio

Di seguito sono presentati il contesto e le fondamenta da cui è scaturita questa tesi. Inoltre viene descritto il dataset utilizzato per la valutazione delle soluzioni individuate e gli specifici obiettivi che si vogliono raggiungere.

2.1 Contesto

Le malattie rare pongono particolari sfide a pazienti, famiglie, caregiver, medici e ricercatori. Vengono considerate "rare" quelle malattie che hanno, nella popolazione generale, una prevalenza inferiore ad una data soglia, codificata dalla legislazione di ogni singolo paese[1]. Dopo essere state trascurate per molti anni (arrivando ad acquisire la denominazione "health orphans"), oggi costituiscono un importante problema di salute pubblica. Le oltre 7000 malattie rare conosciute colpiscono più di 300 milioni di persone nel mondo, il 50% delle quali sono bambini in età pediatrica. Inoltre, la quasi totalità delle malattie rare non ha cura e soltanto il 5% di esse ha opzioni terapeutiche. Se il numero di vite influenzate da una singola malattia rara è per definizione limitato, non lo è la quantità di malattie e di conseguenza l'impatto che hanno sulla popolazione mondiale.

A causa della scarsa disponibilità di informazioni e della loro frammentazione, negli ultimi anni si è vista una forte crescita di comunità di pazienti sul web. Il bisogno di dialogare con altre persone aventi la medesima problematica è spesso una naturale conseguenza della volontà di abbattere le barriere di isolamento in cui si è rinchiusi. I contesti social (quali i gruppi *Facebook*) divengono pertanto il territorio attraverso cui si condividono esperienze, si richiedono pareri e si scambiano informazioni di indubbia rilevanza durante tutto il percorso di un

[1] Ad esempio nell'Unione Europea è considerata rara ogni patologia che colpisce non più di 5 persone ogni 10.000.

malato raro. Purtroppo però spesso rimangono all'interno di questo contesto informale e non vengono comunicate direttamente ai medici: catturare la "voce dei pazienti" può essere un grande passo verso una maggiore inclusione e consapevolezza da parte di medici e ricercatori riguardo alla visione che i diretti interessati hanno della malattia.

2.2 POIROT

Nonostante le recenti novità nell'ambito del *natural language understanding*, la descrizione di fenomeni da documenti testuali è ancora un problema complesso e poco affrontato. Scoprire le ragioni che spieghino un determinato fenomeno espresso nel testo richiede il riconoscimento delle interazioni semantiche tra un insieme di concetti di interesse. I metodi esistenti in NLP solitamente richiedono dati etichettati e usano modelli *black-box* non interpretabili progettati per essere applicati a uno specifico dominio.

POIROT [80, 81, 82] è una metodologia per la spiegazione di fenomeni dal testo: in particolare è progettato per fornire risultati accurati e interpretabili in un contesto non supervisionato, quantificando la loro significatività statistica. Può essere utilizzato sia in modo completamente automatico, sia in maniera parzialmente supervisionata in modo da combinare la conoscenza umana (ad esempio di esperti del settore) con l'analisi automatica del testo. Inoltre è indipendente dallo specifico dominio di utilizzo e dalla lingua dei documenti.

2.2.1 Metodologia

La soluzione proposta è costituita da vari moduli la cui implementazione può essere adattata allo specifico problema considerato. In particolare lo scopo è la costruzione di un *language model* per la rappresentazione del testo in uno spazio semantico latente, che permette di identificare e quantificare relazioni semantiche fra termini, documenti e classi. Viene fatto un uso originale di *Latent Semantic Analysis* (LSA) e *Latent Dirichlet Allocation* (LDA) per l'implementazione di un *language model* algebrico e probabilistico.

La generazione di spiegazioni testuali per un dato fenomeno di interesse viene eseguita incrementalmente. Inizialmente vengono identificate aree dello spazio latente con concentrazione particolarmente alta legate a uno specifico fenomeno di interesse, e viene costruita una spiegazione testuale interpretabile che viene via via raffinata e accompagnata da accurate informazioni probabilistiche.

Per valutare i risultati, la metodologia è stata applicata in ambito medico per catturare la "voce dei pazienti", cioè un insieme di problematiche, esperienze e sentimenti spesso non riportati ai medici, condivisi tramite *post* su *social*

network da una comunità di persone colpite da una specifica malattia rara, l'*acalasia esofagea*. Partendo con solamente brevi *post* non etichettati, POIROT è stato in grado di identificare automaticamente correlazioni scientifiche con un valore di F_1-*score* del 79% e ha fornito spiegazioni valide delle opinioni dei pazienti riguardo alcune delle tecniche disponibili per il trattamento chirurgico della malattia. In generale questa metodologia, applicabile ad altre malattie ma anche a domini diversi, è stata in grado di costruire spiegazioni utili sul punto di vista dei pazienti riguardo argomenti come sintomi, trattamenti, medicinali e alimenti.

2.2.2 Applicazione a *Knowledge Graph Learning*

Rappresentare i risultati ottenuti da POIROT nella forma di *knowledge graph* aumenta significativamente l'espressività, l'interrogabilità e l'interpretabilità della conoscenza estratta. È stata quindi proposta una estensione della metodologia [83, 84] applicabile al problema di *knowledge graph learning* (precedentemente esposto nella Sezione 1.3.2).

L'output non è più una sequenza o un insieme di gruppi composti da termini correlati, ma un *knowledge graph*. Viene aggiunto un passaggio ulteriore per il riconoscimento e la classificazione delle entità menzionate tramite uno strumento di *named entity recognition* (NER). L'inclusione dei termini riconosciuti come entità in una tassonomia gerarchica significa che le correlazioni individuate da POIROT non sono più indipendenti, ma interconnesse tra di loro, e possono essere aggregate in un'unica struttura a grafo. Le nuove informazioni consentono una maggiore interpretabilità, permettendo anche agli utenti meno esperti di comprendere il significato di un certo termine (ad esempio "poem is_a /medicine/surgical_technique"). Inoltre diventa possibile individuare le correlazioni tra due o più classi di entità (ad esempio "farmaco → sintomo", "sintomo → alimento") senza doverle verificare individualmente e senza la necessità di conoscere tutti i termini appartenenti alle classi considerate.

2.2.3 Dataset utilizzato

Il caso di studio su cui POIROT è stato valutato era incentrato sullo studio della Acalasia Esofagea (ORPHA:930). In collaborazione con l'*Associazione Malati Acalasia Esofagea (AMAE)*[2][3], la principale organizzazione di pazienti di questa malattia, sono stati scaricati documenti testuali anonimi dal gruppo

[2] `https://www.amae.it/`

[3] `https://www.orpha.net/consor/cgi-bin/SupportGroup_Search.php?lng=EN&data_id=106412`

Facebook direttamente gestito da loro[4]. Il dataset consiste di 6,917 post e 61,692 commenti diretti (non risposte di altri commenti), pubblicati tra il 21/02/2009 e il 05/08/2019 da circa 2,000 utenti.

In particolare nel corso di questa tesi è stata utilizzata una versione in inglese del dataset originale. Dopo alcune valutazioni è stato scelto di effettuare la traduzione con *Google Translate*.

2.2.4 Limiti

Utilizzando POIROT è possibile sapere quali concetti sono semanticamente correlati e quantificare statisticamente questa correlazione. Non è però possibile risalire a quale sia il significato della relazione: ad esempio è in grado di dire che un certo farmaco è strettamente correlato a un dato sintomo, ma non se il farmaco *allevia* o *aggrava* il sintomo.

Gli archi del *knowledge graph* costruito sulle relazioni estratte da POIROT sono "piatti" perché non possiedono alcuna semantica specifica. Per interpretare correttamente i risultati è perciò necessario possedere una adeguata conoscenza del dominio.

2.3 Obiettivi

Gli obiettivi di questa tesi sono quindi quelli di sopperire ai limiti di POIROT, in particolare **individuando** la semantica delle relazioni stabilite. Inoltre si è interessati ad **aggregare** le relazioni estratte, sia per collassarle in un unico *knowledge graph*, sia per **quantificare** la confidenza con cui sono state scelte anche in base a statistiche estratte dal corpo di testo (ad esempio il numero di volte che la relazione viene espressa nel corpo di testo), per mantenere l'interpretabilità della soluzione.

Più in generale si mira ad estrarre le interazioni semantiche espresse nel testo non etichettato e di aggregarle tra di loro per meglio rappresentare la conoscenza espressa dai pazienti. In un secondo momento le informazioni così ottenute possono essere combinate con quelle proveniente da POIROT per ottenere un *knowledge graph* completo.

Da un punto di vista implementativo sono auspicabili alcune caratteristiche.

- **End-to-end**: nonostante sia probabilmente impossibile utilizzare un unico modello che svolga tutte le funzioni necessarie, si desidera ridurre al minimo il numero di componenti distinte in modo da facilitare l'addestramento e limitare errori legati all'applicazione in *pipeline*.

[4] `https://www.facebook.com/groups/36705181245/`

- **Non supervisionato**: la mancanza di dati etichettati nel dataset utilizzato rende difficoltoso un eventuale addestramento. Un modello addestrabile con dataset di terze parti e poi applicabile a questo specifico caso di studio è comunque molto valido.

Capitolo 3

Soluzioni esistenti

In questo capitolo si descrivono le principali soluzioni prese in esame, con una spiegazione degli approcci proposti e alcune considerazioni sulla loro utilità nello specifico caso di studio.

3.1 Language Models are Open Knowledge Graphs

Il lavoro *Language Models are Open Knowledge Graphs* [15] si pone l'obiettivo di costruire un *knowledge graph* in modo completamente non supervisionato, partendo da documenti testuali in linguaggio naturale e sfruttando la conoscenza contenuta nei grandi *language model*, basati su architetture con miliardi di parametri e addestrati su milioni di documenti testuali.

È importante sottolineare che questo articolo era candidato per partecipare alla conferenza ICLR 2021[1], ma durante la stesura di questa tesi è stato respinto in quanto non sufficientemente specifico. Nei commenti i revisori evidenziano che la sua posizione si trova a metà tra la costruzione di *knowledge graph* e l'analisi della conoscenza contenuta all'interno dei grandi *language model*, limitando i contributi in entrambe le direzioni.

3.1.1 Metodo proposto

Il metodo descritto ha l'obiettivo di estrarre le entità e le relazioni contenute in un testo generico, semplicemente con l'utilizzo di un *language model* preaddestrato (senza la necessità di effettuare *fine-tuning*) ed è indipendente dal modello utilizzato (ad esempio BERT, GPT, T5...). L'approccio proposto,

[1] `https://iclr.cc/`

Figura 3.1: Le fasi principali di *Language Models Are Open Knowledge Graphs*.
Fonte: `https://towardsdatascience.com/language-models-are-open-knowledge-graphs-but-are-hard-to-mine-13e128f3d64d`

chiamato MaMa e delineato in Figura 3.1, è composto da due fasi principali, *Match* e *Map*, oltre che da una di pre-elaborazione del testo.

Fase di pre-elaborazione

In questa prima fase, i documenti vengono suddivisi in frasi che saranno poi analizzate singolarmente, e vengono estratti i *noun chunks* (o *noun phrases*, gruppi nominali in italiano)[2]. I *noun chunks* verranno usati come entità per creare le triple. Entrambi i compiti vengono effettuati utilizzando la libreria **spaCy**[3], che offre strumenti per il NLP allo stato dell'arte comodamente utilizzabili in Python.

Match

I grandi *language model* contengono conoscenza generica e di ampio spettro, appresa dagli enormi dataset su cui sono stati addestrati, che non corrisponde esattamente a quella espressa dal testo analizzato. Lo scopo di questa fase

[2]Sono porzioni di testo che hanno lo stesso ruolo di un singolo sostantivo, e che quindi possono essere sostituite con un pronome per ottenere una frase comunque sensata; oltre al sostantivo vero e proprio comprendono anche articoli, aggettivi e altri termini che ne modificano il significato; ad esempio *"Il grillo di Pinocchio è saggio"* può diventare *"Esso è saggio"* rimanendo corretta.

[3]`https://spacy.io`

è quello di mettere in relazione la conoscenza del *language model* con i fatti contenuti nel testo, per ottenere una serie di triple (chiamate "fatti candidati") nella forma *(testa, relazione, coda)* in cui *testa* e *coda* sono entità collegate dalla relazione indicata.

Ogni frase viene analizzata singolarmente dal *language model*, senza aver necessariamente effettuato alcun *fine-tuning*, per ottenere una matrice di *attention* tra i termini che contiene. Dato che i *language model* utilizzati sono formati da più livelli in serie (vedi Sezione 1.1), gli autori hanno effettuato esperimenti considerando i valori di *attention* solo dell'ultimo livello, oppure la media dei valori di tutti i livelli: la prima soluzione è risultata più efficace, probabilmente perché nei primi livelli vengono apprese strutture linguistiche più basilari che non contengono conoscenza.

A questo punto per ogni possibile coppia di entità (testa e coda) contenute nella frase (identificate dai *noun chunks* nella fase di pre-elaborazione) viene generato un fatto candidato. Si considera **solamente il testo che si trova fra le due entità**, da cui vengono selezionate alcune parole che descrivono la relazione che le lega, sulla base dei valori di *attention*. Viene usato un processo iterativo che, partendo dalla *testa* e fermandosi quando si arriva alla *coda*, estrae una parola alla volta: per selezionare la prossima parola si valutano i valori di *attention* della parola corrente verso tutte le altre e si sceglie quella con valore più alto (esempio in Figura 3.2). Una volta raggiunta la *coda* il processo termina e la relazione è la concatenazione delle parole selezionate. Alla tripla (fatto) così ottenuta viene inoltre associato un **valore di confidenza**, calcolato come somma dei valori di *attention* delle parole selezionate. Se una delle due entità è un pronome, viene usata la libreria **neuralcoref**[4] per sostituirlo con il *noun chunk* a cui si riferisce. Questo algoritmo di selezione è stato implementato realizzando un grafo che connette la parole, in cui il peso degli archi è il valore di *attention* tra le due parole; viene usato un algoritmo di tipo *beam search* per costruire un percorso tra *testa* e *coda*, che contiene le parole selezionate per la relazione.

Infine i fatti candidati vengono **filtrati** per rimuovere quelli meno utili, secondo alcuni criteri.

1. Il valore di confidenza deve essere maggiore di una soglia prefissata: in caso contrario il fatto non rispecchia efficacemente la conoscenza contenuta nel *language model*.

2. La frequenza della relazione nell'intero dataset deve essere oltre una soglia prefissata: questo permette di rimuovere le relazioni troppo specifiche, e probabilmente poco significative, che compaiono solo pochissime volte all'interno del testo.

[4] `https://github.com/huggingface/neuralcoref`

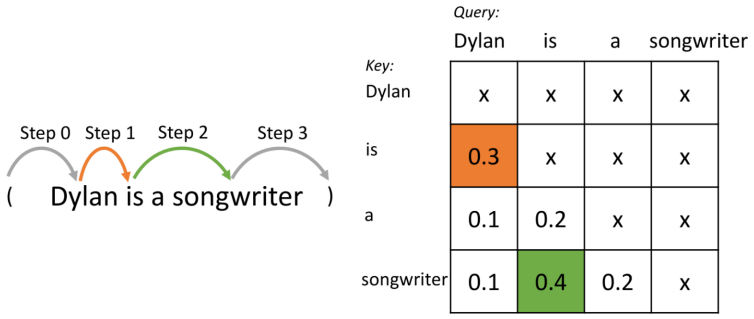


Figura 3.2: Processo iterativo per l'estrazione del predicato: partendo dalla *testa* per arrivare alla *coda*, si scelgono le parole in base ai valori di *attention*. Fonte: [15]

3. La relazione deve essere formata da parole che nel testo compaiono consecutivamente: in caso contrario non avrebbe alcuna interpretazione significativa.

I fatti che non soddisfano tutti e tre i vincoli vengono eliminati.

Map

Nell'ultima fase i componenti dei fatti candidati (*testa*, *relazione* e *coda*) vengono messi in corrispondenza con lo schema di una *knowledge base* definita esternamente, ad esempio quella di Wikidata.

Per quanto riguarda le entità, il problema di *entity linking* è stato trattato nella sezione 1.3.2. Gli autori si limitano a dire di utilizzare un modello non supervisionato, che sfrutta gli *embedding* delle parole nel contesto della frase per risolvere le ambiguità.

Per le relazioni viene utilizzato un procedimento più complesso. Inizialmente vengono normalizzate, rimuovendo verbi ausiliari, aggettivi, avverbi e applicando la lemmatizzazione[5]. Poi viene usata un'euristica basata sulle co-occorrenze delle entità nelle triple per costruire una prima mappatura tra le relazioni estratte e quella espresse nella *knowledge base*, che viene controllata manualmente da uno degli autori per rimuovere eventuali risultati sbagliati[6].

È possibile che per alcune entità o relazioni non venga trovata alcuna corrispondenza: vengono comunque conservate in quello che viene chiamato *schema aperto*. Se nessuna delle componenti di un fatto è stata mappata si dice che è esso "completamente non mappato", altrimenti si parla di fatto "parzialmente mappato". In ogni caso vengono memorizzati nello "schema

[5]Lemmatizzare in questo contesto significa sostituire un termine con la sua forma base (il lemma): ad esempio i verbi all'infinito, i nomi al singolare maschile...

[6]Gli autori dicono che questa operazione ha richiesto solo un giorno.

aperto": l'idea è quella di essere in grado di trovare nuove relazioni o entità per arricchire ulteriormente la *knowledge base* utilizzata.

3.1.2 Considerazioni

La sua semplicità e il fatto di essere completamente non supervisionato sono stati i motivi per cui questo lavoro è stato considerato. Il *knowledge graph* risultante può essere poi integrato con quello costruito da POIROT, selezionando solo le entità di interesse per estrarre le relazioni tra di esse.

La principale nota negativa è la struttura a *pipeline*: vengono usati modelli esterni sia per l'identificazione delle entità (come *noun chunks*) sia per la risoluzione delle coreferenze.

3.2 ATLOP

Il paper *"Document-Level Relation Extraction with Adaptive Thresholding and Localized Context Pooling"* [17] propone un nuovo metodo per l'estrazione di relazioni da testo in linguaggio naturale. Risolve il problema di estrarre le relazioni da un intero documento, che pone numerose sfide rispetto a lavorare a livello di frase. Nel secondo caso, più trattato in letteratura, si hanno solitamente semplici frasi contenenti una singola coppia di entità e che esprimono un'unica relazione; al contrario a livello di documento possono essere presenti molte coppie di entità e per ognuna di esse possono sussistere molteplici relazioni.

Per risolvere il problema viene impiegato un modello con due passaggi: inizialmente tramite un *language model* si ottengono gli *embedding* delle entità, poi per ogni coppia un classificatore multilabel si occupa di determinare quali relazioni sussistono.

3.2.1 Metodo proposto

Sulla base della struttura a due passaggi, gli autori propongono anche due nuove tecniche per risolvere alcune limitazioni riscontrate in altre soluzioni. È importante notare che si parte dal presupposto che le menzioni siano già state identificate nel testo, e che siano state associate alla giusta entità a cui fanno riferimento.

- **Localized Context Pooling** Gli *embedding* ottenuti dovrebbero catturare anche il contesto relativo all'entità. È comune però che ogni entità compaia in più di una coppia: usando in tutte la stessa rappresentazione viene preso in considerazione anche contesto irrilevante per la specifica coppia, introducendo del rumore che può penalizzare i risultati del

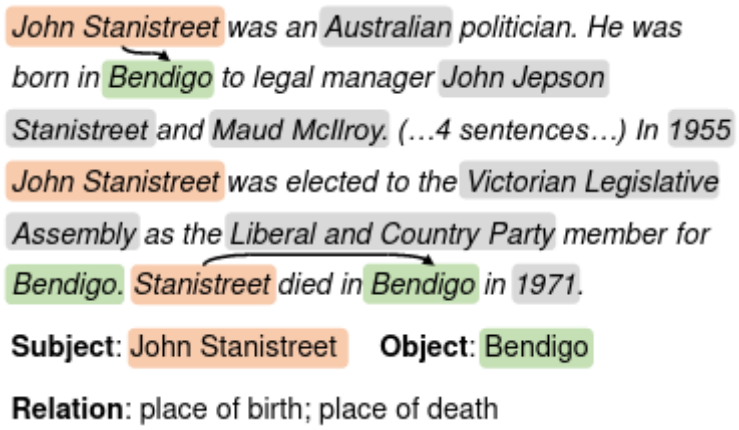


Figura 3.3: Un esempio di documento contenente molteplici entità e relazioni. Il soggetto *"John Stanistreet"* (in arancio) e l'oggetto *"Bendigo"* (in verde) esprimono le relazioni *luogo di nascita* e *luogo di morte*. Le menzioni interessate sono collegate da una freccia. Le altre entità sono evidenziate in grigio.
Fonte: [17]

classificatore. Per questo si propone di utilizzare i valori di *attention* del *language model* per determinare il contesto comune alle due entità contenute nella coppia.

- **Adaptive Threshold** L'output del classificatore multilabel utilizzato per determinare le relazioni è un valore di probabilità per ogni tipologia di relazione. Solitamente viene applicata una soglia fissa, scelta empiricamente, per determinare quali classi sono positive (tutte quelle con probabilità maggiore della soglia). Questo può introdurre un certo errore, perché la soglia migliore sul dataset usato in fase di sviluppo potrebbe non essere ottimale per tutte le istanze. Si propone quindi di utilizzare un valore addestrabile che venga determinato dal modello e possa essere utilizzato come soglia dinamica per dividere le classi positive e negative.

L'approccio proposto è delineato in Figura 3.4 e descritto di seguito.

Language model

Il primo passaggio ha lo scopo di ottenere gli *embedding* per le entità, usando un *language model* preaddestrato (in particolare BERT) senza necessità di effettuare *fine-tuning*. Il testo originale viene modificato, aggiungendo il simbolo speciale "*" all'inizio e alla fine di ogni menzione delle entità. Viene quindi analizzato da BERT per ottenere gli *embedding*: per ogni menzione

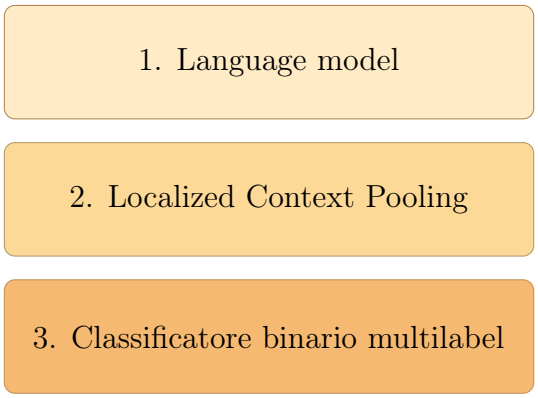


Figura 3.4: Le fasi principali del modello ATLOP

viene considerato l'*embedding* del simbolo "*" posto all'inizio della menzione stessa. Per ogni entità e_i viene utilizzato *logsumexp*[7] tra gli *embedding* delle menzioni per calcolarne uno unico $\mathbf{h}_{e_i}$.

$$\mathbf{h}_{e_i} = log \sum_j exp\left(\mathbf{h}_{m_j^i}\right) \tag{3.1}$$

dove $\mathbf{h}_{m_j^i}$ è l'*embedding* della j-esima menzione dell'entità e_i.

Localized Context Pooling

Con l'Equazione 3.1 vengono combinati tutti gli *embedding* delle menzioni di una certa entità, per ottenerne uno unico che cattura anche tutti i contesti in cui compaiono le menzioni. Questo contesto potrebbe non essere rilevante per tutte le coppie che interessano questa entità: ad esempio in Figura 3.3 la seconda menzione di *John Stanistreet* e il suo contesto sono irrilevanti per la coppia *(John Stanistreet, Bendigo)*. Per determinare il contesto comune ad entrambe le entità viene sfruttato il livello di *attention* del *language model*. Dalla matrice di *attention* $A \in \mathbb{R}^{H \times l \times l}$ tra tutti i token (dove A_{ijk} è il l'attention tra il token j e il token k nella i-esima *head*) viene calcolata una nuova matrice per ogni entità $A_i^E \in \mathbb{R}^{H \times l}$ facendo la media delle *attention* delle menzioni. Poi data una coppia (e_s, e_o) viene identificato il contesto comune facendo il prodotto elemento per elemento tra le due matrici (calcolando la media delle *head*) e moltiplicandola con la matrice degli *embedding* dell'intero documento. Si ottiene così un vettore $\mathbf{c}^{(s,o)}$ (*embedding*) che codifica il contesto comune alle due entità della coppia.

[7] Una versione "smussata" della funzione massimo

Classificatore binario multilabel con adaptive thresholding

Infine gli *embedding* ottenuti vengono usati con un classificatore binario multilabel per determinare quali relazioni sussistono tra la coppia considerata. Dati gli *embedding* $(\mathbf{h}_{e_s}, \mathbf{h}_{e_o})$ per una coppia e_s, e_o e il vettore per il contesto comune $c^{(s,o)}$, vengono generate due nuove rappresentazioni $\mathbf{z}$ delle entità, applicando prima una trasformazione lineare seguita da una funzione non lineare, poi usate per calcolare la probabilità della relazione r tramite funzione bilineare e funzione sigmoide:

$$\begin{aligned} z_s^{(s,o)} &= \tanh\left(\mathbf{W}_s \mathbf{h}_{e_s} + \mathbf{W}_{c_1} \mathbf{c}^{(s,o)}\right), \\ z_o^{(s,o)} &= \tanh\left(\mathbf{W}_o \mathbf{h}_{e_o} + \mathbf{W}_{c_2} \mathbf{c}^{(s,o)}\right), \\ \Pr(r|e_s, e_o) &= \sigma(z_s^T \mathbf{W}_r z_o + b_r) \end{aligned} \tag{3.2}$$

dove $\mathbf{W}_s, \mathbf{W}_o, \mathbf{W}_{c_1}, \mathbf{W}_{c_2}, \mathbf{W}_r \in \mathbb{R}^{d \times d}$ e $b_r \in \mathbb{R}$ sono parametri del modello.

Per realizzare la soglia dinamica viene aggiunga una nuova classe TH, predetta come altre dall'Equazione 3.2. Durante l'addestramento, data una coppia $T = (e_s, e_o)$, le relazioni vengono divise in due insiemi $\mathcal{P}_T$ e $\mathcal{N}_T$, rispettivamente per le classi positive (le relazioni che esistono per T) e negative (quelle che non esistono). Viene poi usata una *ranking loss* per rendere maggiori di TH le predizioni delle classi positive, e minori quelle delle classi negative.

$$\begin{aligned} \mathcal{L}_1 &= -\sum_{r \in \mathcal{P}_T} \log\left(\frac{\exp(\text{logit}_r)}{\sum_{r' \in \mathcal{P}_t \cup \{TH\}} \exp(\text{logit}_{r'})}\right), \\ \mathcal{L}_2 &= -\log\left(\frac{\exp(\text{logit}_{TH})}{\sum_{r' \in \mathcal{N}_T \cup \{TH\}} \exp(\text{logit}_{r'})}\right), \\ \mathcal{L} &= \mathcal{L}_1 + \mathcal{L}_2. \end{aligned} \tag{3.3}$$

La prima parte $\mathcal{L}_1$ interessa solo le classi positive e TH: viene usata la *categorical cross entropy* dato che potrebbe esserci più di una classe positiva. La seconda parte $\mathcal{L}_2$ comprende solo le classi negative e TH, e viene usata la *cross entropy* in cui TH è l'unica etichetta corretta. Le due parti sono semplicemente sommate per ottenere la *loss* totale.

In fase di inferenza vengono considerate classi "positive" tutte quelle la cui predizione è maggiore di quella di TH.

3.2.2 Modelli preaddestrati

Gli autori hanno rilasciato due modelli preaddestrati, uno basato su BERT-LARGE e uno su RoBERTa-BASE. Il dataset utilizzato, DocRED [85], contiene

circa 5000 documenti presi da Wikipedia. Le tipologie di relazioni sono relativamente poche (96) e tutte molto generiche, applicabili perlopiù ad entità come personaggi famosi, elementi geografici o politici.... I dati di interesse riguardano invece un contesto quotidiano, ma con terminologia relativa a un dominio specifico, quindi probabilmente sarà necessario riaddestrare un modello su dati più in linea con il nostro caso di studio. Purtroppo i dataset per l'estrazione di relazioni a livello di documento sono estremamente limitati.

3.2.3 Considerazioni

Questa soluzione si limita all'estrazione di relazioni, che è l'obiettivo più importante e impegnativo della tesi. La capacità di catturare interazioni a livello di documento piuttosto che di frase la rende molto promettente. Le relazioni ottenute possono essere integrate direttamente nel *knowledge graph* costruito a partire dalle correlazioni estratte da POIROT. Sarà però necessario identificare le entità con qualche strumento esterno, introducendo di conseguenza un errore più o meno considerevole.

3.3 DeepEventMine

Il lavoro *DeepEventMine: end-to-end neural nested event extraction from biomedical texts* [13] descrive una metodologia basata, anche in questo caso, sull'uso dei *language model* preaddestrati, per il compito dell'*event extraction* (vedi Sezione 1.2.1). La metodologia proposta supera alcuni limiti spesso presenti in altre soluzioni. DeepEventMine è in grado di gestire eventi innestati (in cui gli argomenti di un evento possono essere a loro volta altri eventi), e non solo eventi "piatti" (non innestati), come invece considerati dalla maggior parte dei lavori noti in letteratura. Inoltre spesso si dà per scontato che le entità siano state precedentemente identificate nel testo, mentre in questo caso vengono estratte contemporaneamente agli eventi da un unico modello *end-to-end*.

3.3.1 Metodo proposto

Il modello viene utilizzato specificatamente nel dominio biomedico, ma può essere direttamente applicato anche a qualsiasi altra situazione. La sua struttura (delineata in Figura 3.5) è composta da quattro livelli: il primo è formato da un *language model* preaddestrato (in questo caso SciBERT); il secondo si occupa di identificare le entità e i *trigger* e assegnare loro una classe; nel terzo viene determinato il ruolo dei possibili argomenti (siano essi entità o

Figura 3.5: Struttura di DeepEventMine

trigger) rispetto ai *trigger*; infine nell'ultimo livello vengono costruiti i possibili eventi, e scartati quelli non validi.

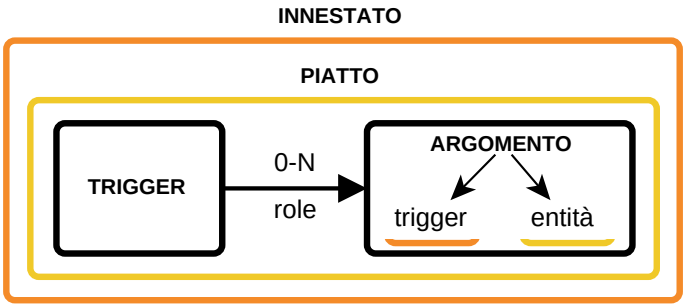


Figura 3.6: Ad ogni *trigger* è associato un numero variabile di argomenti, ognuno dei quali può essere un'entità o un altro *trigger*: se gli argomenti sono tutti entità l'evento è "piatto", altrimenti è innestato.

Di seguito i passaggi sono descritti nello specifico.

BERT

Il testo viene inizialmente suddiviso in frasi, e ogni frase è processata separatamente. Utilizzando il *language model* si ottengono gli *embedding* dei singoli token. Si assume che la frase S abbia n parole, e che l'i-esima parola

venga spezzata in s_i token. Questo livello assegna un *embedding* $\mathbf{v}_{i,j}$ al j-esimo token dell'i-esima parola, e calcola un vettore $\mathbf{v}_S$ per l'intera frase.

Estrazione di entità e *trigger*

Questo livello si occupa di identificare le entità e i *trigger* nel testo, e di assegnare loro la classe a cui appartengono. Vengono valutati tutti i possibili intervalli di parole nella frase che sono più corti di un valore prefissato L_{ent} o L_{trig}, rispettivamente per le entità e i *trigger*. Per ognuno viene calcolata una nuova rappresentazione $\mathbf{m}_{k,l} \in \mathbb{R}^{d_m}$ combinando gli *embedding* dei token dal k-esimo al l-esimo:

$$\mathbf{m}_{k,l} = \left[\mathbf{v}_{k,1}; \frac{\sum_{i=j}^{l} \sum_{j=1}^{s_i} \mathbf{v}_{i,j}}{\sum_{i=k}^{l} s_i}; \mathbf{v}_{l,s_l} \right] \tag{3.4}$$

dove $[;;]$ denota la concatenazione. Vengono concatenati tre componenti: l'*embedding* del primo *token* dell'intervallo; la media degli *embedding* di tutto l'intervallo; l'*embedding* dell'ultimo token.

Le rappresentazioni così ottenute vengono analizzate da un classificatore multilabel che determina se si tratta di una entità o di un *trigger*, e eventualmente a quale classe appartiene.

Ruoli

Ora vengono considerate tutte le possibili coppie *trigger*-argomento per determinare il ruolo che potrebbe ricoprire. Gli *embedding* $\mathbf{m}_t$ calcolati nell'Equazione 3.4 vengono completati concatenandoli con una rappresentazione vettoriale della classe predetta nel livello precedente ($\mathbf{s}_t$) ottenendo un nuovo vettore $\mathbf{v}_t$ (se per un intervallo sono state determinate più classi, vengono considerate come entità differenti).

$$\mathbf{v}_t = [\mathbf{m}_t; \mathbf{s}_t] \,. \tag{3.5}$$

Questo viene fatto sia per il *trigger* ($\mathbf{v}_t$) sia per l'argomento ($\mathbf{v}_a$).

Viene calcolata una rappresentazione per l'intera coppia $\mathbf{r}_i \in \mathbb{R}^{d_r}$ come

$$\mathbf{r}_i = \mathrm{GELU}\left(\mathbf{W}_r \left[\mathbf{v}_t; \mathbf{v}_a; \mathbf{c}\right] + \mathbf{b}_r\right) \tag{3.6}$$

dove $\mathbf{W}_r$ e $\mathbf{b}_r$ sono parametri del modello, e $\mathbf{c} = \mathbf{v}_S$ è l'*embedding* della frase ottenuto dal primo livello. Un classificatore multi-classe si occupa di individuare il giusto ruolo per la i-esima coppia utilizzando $\mathbf{r}_i$.

Eventi

Nell'ultimo livello vengono creati gli eventi veri e propri. Partendo dai risultati precedenti, viene costruita una serie di eventi candidati combinando i *trigger* e gli argomenti, per poi scartare quelli non validi.

Di ogni evento viene calcolata una rappresentazione vettoriale combinando quella del *trigger* ($\mathbf{v}_t$) e del ruolo ($\mathbf{r}_i$) dal terzo livello, e quelle degli argomenti. Questa viene poi usata da un classificatore binario per determinare se l'evento è valido o meno.

Per gestire eventi innestati, vengono classificati prima quelli in cui gli argomenti sono tutti entità e poi tutti gli altri gerarchicamente.

Infine gli eventi validi vengono analizzati da un altro classificatore multilabel per determinare eventuali modificatori (ad esempio la negazione).

3.3.2 Modelli preaddestrati

DeepEventMine ha ottenuto lo stato dell'arte in **sette** dataset di dominio biomedico: *"Cancer Genetics"* (CG), *"GENIA Event Extraction, 2011"* (GE11), *"GENIA Event Extraction, 2013"* (GE13), *"Infectious Diseases"* (ID), *"Epigenetics and Post-translational Modifications"* (EPI), *"Pathway Curation"* (PC) e *"Multi-Level Event Extraction"* (MLEE). Per ognuno è stato addestrato un modello dedicato, tutti disponibili pubblicamente. Per tutti i sette modelli i risultati sono ottimi sul dataset con cui sono stati addestrati, ma diventano rapidamente scadenti appena ci si allonana anche poco dal loro dominio specifico. Il codice per l'addestramento non è ancora stato rilasciato[8], per cui al momento non è possibile utilizzare un proprio dataset. Inoltre la disponibilità di dataset con annotazioni di eventi è molto scarsa, in particolar modo fuori dal contesto biomedico.

3.3.3 Considerazioni

La rappresentazione delle interazioni sotto forma di eventi innestati permette di catturarle con un'elevata espressività e ottenere un risultato molto strutturato, utile a una successiva elaborazione. Inoltre l'estrazione contemporanea delle entità rimuove la necessità di uno strumento esterno: questo permette si un addestramento più mirato all'identificazione di eventi, sia una maggiore flessibilità dato che tutti i compiti vengono ottimizzati sul dominio di interesse.

D'altro canto la limitata generalizzazione dei modelli, insieme alle difficoltà nell'addestramento esposte nella sezione precedente, possono renderne problematico l'utilizzo.

[8]Lo sarà probabilmente in futuro.

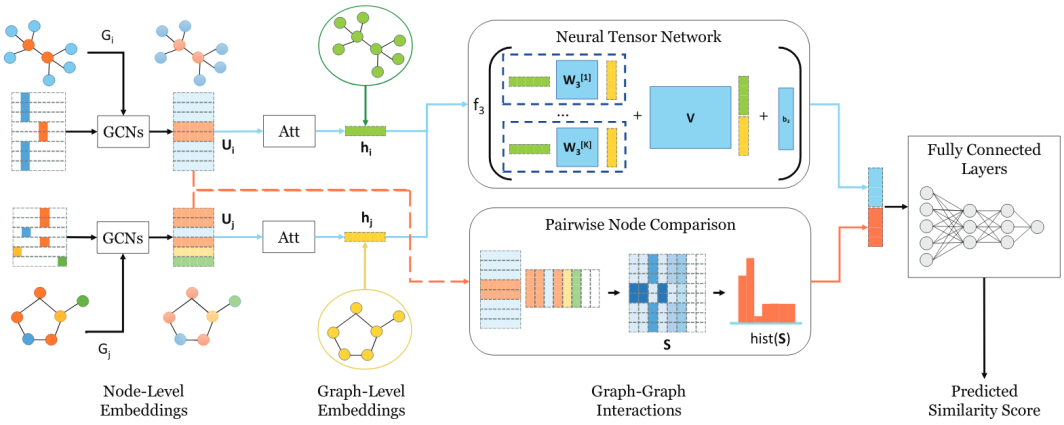


Figura 3.7: La struttura del modello SimGNN
Fonte: [66]

3.4 SimGNN

Il lavoro *SimGNN: A Neural Network Approach to Fast Graph Similarity Computation* [66] si dà l'obiettivo di calcolare un valore di similarità tra grafi utilizzando un approccio neurale. Gli algoritmi per il calcolo della similarità tra grafi (ad esempio *Graph Edit Distance* o *Maximum Common Subgraph*) hanno un elevato costo computazionale che li rende insostenibili per grafi con più di 16 nodi. Viene perciò proposto un approccio neurale che riformula l'obiettivo come problema di apprendimento, progettato per soddisfare alcuni requisiti.

- **Indipendente dalla rappresentazione** Lo stesso grafo può essere rappresentato con diverse matrici di adiacenza tramite permutazione dei nodi; il punteggio di similarità deve essere indipendente da queste trasformazioni.
- **Induttivo** Il calcolo della similarità deve essere generalizzato anche a grafi mai visti, fuori dal *training set*.
- **Apprendibile** Il modello deve essere adattabile ad ogni metrica di similarità fornitagli durante l'addestramento.

La soluzione è composta da due diversi approcci, che vengono usati assieme: nel primo vengono utilizzati gli *embedding* degli interi grafi, mentre il secondo confronta direttamente gli *embedding* dei singoli nodi.

3.4.1 Metodo proposto

Il modello proposto (illustrato in Figura 3.7) riceve in input una coppia di grafi e dà come risultato un punteggio nell'intervallo $[0, 1]$ che indica quanto

sono simili. Inizialmente si calcolano gli *embedding* dei nodi di entrambi i grafi utilizzando delle *Graph Convolutional Network* (GCN), che vengono poi utilizzati parallelamente per due diverse strategie. La prima li combina usando un componente di *attention* in modo ottenere un *embedding* per ogni grafo, poi confrontati con una *Neural Tensor Network*. La seconda strategia calcola la similarità di ogni coppia di nodi e realizza un istogramma con i valori così ottenuti. I risultati delle due strategie vengono concatenati e analizzati da una rete neurale *feed-forward* per ottenere il punteggio di similarità. È importante sottolineare che solamente la prima viene usata durante l'addestramento, perché il calcolo dell'istogramma non è differenziabile e quindi non può essere usato per la discesa del gradiente.

Di seguito sono descritti nello specifico i singoli passaggi.

Embedding dei nodi

Il primo passaggio è comune ad entrambe le strategie applicate. Gli autori hanno scelto di utilizzare delle GCN in questa fase perché il risultato è indipendente dall'ordine dei nodi (primo requisito) e funzionano anche con grafi mai visti prima (secondo requisito). Su ognuno dei due grafi in esame vengono applicati tre livelli di GCN per ottenere gli *embedding* dei relativi nodi.

Strategia 1: embedding dei grafi

A partire dal risultato precedente e utilizzando un componente di *attention* viene calcolata un'unica rappresentazione vettoriale separatamente per ogni grafo.

In particolare viene creato prima un vettore di riferimento facendo la media tra tutti gli *embedding* dei nodi e applicando una trasformazione lineare seguita da una non lineare. Ogni *embedding* viene poi confrontato con quello di riferimento per ottenere un valore di *attention* specifico, tanto più alto quanto i due sono simili. I coefficienti così ottenuti vengono usati per calcolare una media pesata degli *embedding*; non viene fatta alcuna normalizzazione in modo che la norma degli *embedding* rifletta la dimensione del grafo, che è una importante caratteristica da considerare per determinare la similarità. Lo scopo di questo componente è quello di permettere al modello di "imparare" autonomamente a identificare ed enfatizzare i nodi più significativi.

I due nuovi vettori vengono confrontati fra loro con una *Neural Tensor Network* per ottenere K valori distinti di similarità, dove K è un iperparametro.

Strategia 2: confronto diretto dei nodi

Un unico *embedding* per grafo non è in grado di riflettere tutte le informazioni specifiche sui nodi; spesso le differenze tra due grafi stanno in piccole sottostrutture che è difficile catturare in un'unica rappresentazione vettoriale. Perciò seguendo questa strategia gli *embedding* dei nodi vengono confrontati direttamente per ottenere una matrice di similarità $S \in \mathbb{R}^{N_i \times N_j}$, dove N_i e N_j sono rispettivamente il numero di nodi del primo e del secondo grafo:

$$S = \sigma(U_i U_j) \tag{3.7}$$

dove $U_i \in \mathbb{R}^{N_i \times D}$ e $U_j \in \mathbb{R}^{N_j \times D}$ sono le matrici degli *embedding* dei nodi per i due grafi.

Per far sì che il calcolo rimanga indipendente dall'ordinamento dei nodi, viene estratto l'istogramma da S con B intervalli, $\text{hist}(S) \in \mathbb{R}^B$, dove B è un iperparametro. Si ottiene un vettore che contiene, per ogni intervallo di similarità, la frequenza con cui compare all'interno di S, che codifica quindi una aggregazione delle similarità tra i nodi.

È importante notare che il calcolo dell'istogramma non è derivabile, e non è quindi utilizzabile per l'addestramento. Inoltre, dato che la dimensione della matrice S è quadratica rispetto alla dimensione dei grafi, questa strategia potrebbe diventare molto costosa; è perciò possibile ignorarla se necessario e utilizzare solamente la prima.

Calcolo del punteggio di similarità

Nell'ultimo passaggio i risultati ottenuti con le due strategie vengono concatenati e analizzati da una rete neurale completamente connessa per ottenere il punteggio di similarità finale.

3.4.2 Considerazioni

Il metodo proposto è stato valutato per l'aggregazione di eventi ottenuti utilizzando DeepEventMine. Le *Graph Neural Network* (GNN) (tra cui le *Graph Convolutional Network* usate in questo caso) hanno recentemente dimostrato una grande efficacia per lavorare sui grafi, in particolare quando usate per calcolare gli *embedding* dei nodi o di strutture più complesse. L'approccio che utilizzano (vedi Sezione 1.4.1) permette di ottenere massimo e diretto beneficio dalla struttura del grafo e dalle informazioni in esso contenute.

Purtroppo però presenta numerose limitazioni. È un algoritmo supervisionato, per cui è necessario avere dati etichettati per l'addestramento. Essendo in grado di calcolare la similarità tra una coppia di grafi, non è possibile effettuare

clustering; inoltre a seconda del numero di grafi da valutare, le coppie potrebbero essere molte e quindi richiedere un tempo molto elevato. L'algoritmo considera solamente gli attributi dei nodi, mentre gli archi vengono considerati "piatti": nel caso di DeepEventMine gli archi rappresentano i ruoli coperti dagli argomenti rispetto al *trigger*, per cui si perderebbero informazioni importanti. Infine non viene fornita alcuna mappatura tra i nodi dei due grafi, per cui è impossibile stabilire quali di essi incidono maggiormente sul punteggio finale. Per questi motivi gli esperimenti effettuati sono stati estremamente limitati. Le modifiche effettuate sul codice sono state integrate nel repository ufficiale disponibile su GitHub[9].

3.5 DDGK

In *DDGK: Learning Graph Representations for Deep DivergenceGraph Kernels* [71] viene proposto un approccio completamente non supervisionato per costruire *embedding* di interi grafi. L'idea è quella di definire una funzione kernel e di usarla per rappresentare i grafi in un *Dissimilarity Space* (vedi Sezione 1.4.1).

In primo luogo *DDGK* non fa alcuna assunzione sulla struttura dei grafi di input: propone invece un meccanismo chiamato *isomorphism attention* in grado di allineare i nodi tra due grafi diversi. In secondo luogo non dipende da nessuna euristica precalcolata: la funzione kernel viene appresa contemporaneamente con l'allineamento dei nodi, in modo da lasciare al modello la libertà di imparare la migliore rappresentazione che preservi le caratteristiche dei grafi. Infine, essendo un approccio non supervisionato, si basa solamente sulle somiglianze strutturali e il risultato non dipende da come verranno poi utilizzati gli *embedding* (classificazione, clustering...).

3.5.1 Metodo proposto

Il metodo proposto mira a rappresentare i grafi in un *dissimilarity space*: come delineato nella Sezione 1.4.1 è necessario selezionare un insieme R di prototipi e una funzione per misurare la dissimilarità fra coppie di elementi.

In questo caso vengono utilizzati tre componenti chiave:

- **Codifica dei prototipi** Per ogni prototipo viene addestrato un autoencoder che impara a riprodurre la sua struttura.
- ***Attention* inter-grafo** Questo meccanismo è in grado di imparare gli allineamenti tra i nodi di due grafi; è necessario perché i nodi potrebbero

[9] `https://github.com/benedekrozemberczki/SimGNN`

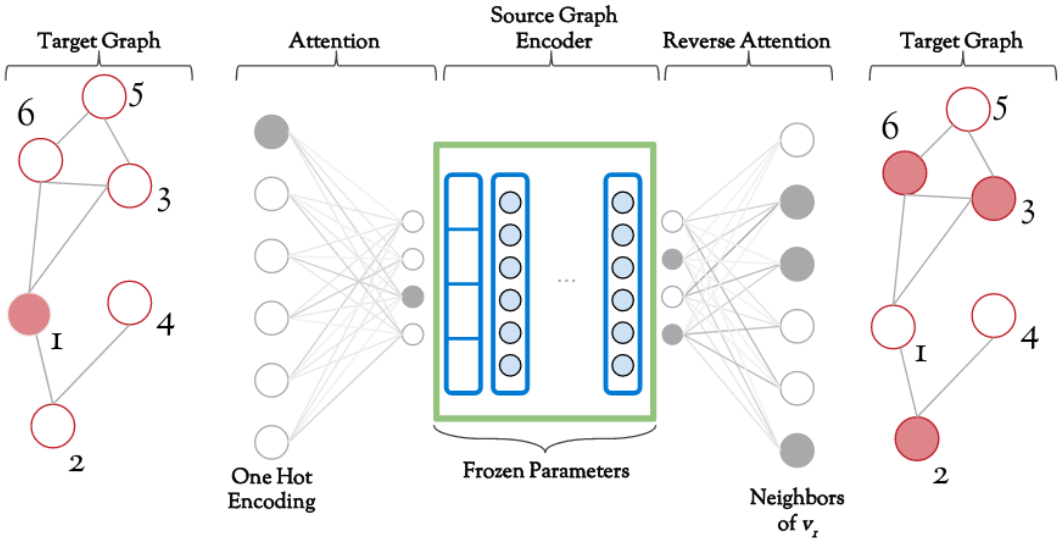


Figura 3.8: Tramite i due livelli di *attention* il modello è in grado di riprodurre la struttura di G_T usando quella di G_S.
Fonte: [71]

avere identificatori diversi, e i due grafi potrebbero anche avere grandezza diversa. Di conseguenza l'allineamento appreso non è necessariamente uno a uno. Viene utilizzato per confrontare ogni grafo con i prototipi e ottenere dei valori di divergenza (dissimilarità).

- **Coerenza degli attributi** Fin'ora si è presa in considerazione solamente la caratteristica struttura dei grafi in esame. Solitamente però i nodi e gli archi possiedono degli attributi che ne specificano la semantica e che giocano un ruolo importante nel determinare la similarità tra i grafi.

La similarità tra una coppia di grafi viene misurata sulla base di quanto efficace sia l'allineamento individuato tramite l'*attention* inter-grafo.

Di seguito vengono descritti i passaggi nello specifico.

Codifica dei prototipi

Per ogni prototipo viene addestrato un autoencoder che impara a riprodurre la matrice di adiacenza. È implementato come classificatore multilabel che, preso in input un nodo, predice i suoi vicini all'interno del grafo interessato.

Attention inter-grafo

Sono dati due grafi, G_T e G_S dove il secondo è uno dei prototipi. Si vogliono addestrare due modelli: il primo ($\mathcal{M}_{T \to S}$) che sia in grado di allineare i nodi

di G_T con quelli di G_S; il secondo ($\mathcal{M}_{S\to T}$) deve imparare ad allineare un gruppo di nodi di G_S con un gruppo di nodi di G_T. Vengono implementati rispettivamente come un classificatore multiclasse e come un classificatore multilabel.

Aggiungendo questi due livelli rispettivamente all'input e all'output dell'autoencoder addestrato su G_S si ottiene un modello che è in grado di riprodurre la struttura di G_T utilizzando quella di G_S. La Figura 3.8 illustra questa configurazione. Durante l'addestramento vengono modificati solamente i parametri dei due livelli di *attention*, quelli dell'autoencoder rimangono invariati.

Coerenza degli attributi

Il meccanismo di *attention* è in grado di imparare un allineamento tra i nodi dei due grafi, ma tiene in considerazione solamente la loro struttura, ignorando gli attributi. Per questo vengono aggiunte due funzioni *loss* specifiche ai livelli di *attention*.

Consideriamo che ogni nodo abbia una etichetta appartenente a un insieme definito Y. Per ogni nodo del grafo target $u_i \in G_T$ è possibile definire una distribuzione di probabilità delle etichette $Q_{T_n} = \Pr(y_j|u_i), y_j \in Y$, che sarà 0 per tutti i valori di j tranne uno, per cui varrà 1. Lo stesso è applicabile per ogni nodo $v_k \in G_S$. Dato che $\mathcal{M}_{T\to S}$ impara una distribuzione $\Pr(v_k|u_i), v_k \in G_S$ è possibile calcolare la distribuzione delle etichette predetta dalla rete come

$$Q_{a_n}(y_j|u_i) = \sum_k \mathcal{M}_{T\to S}(v_k|u_i) \cdot \Pr(y_j|v_k). \tag{3.8}$$

È possibile quindi confrontare Q_{T_n} e Q_{a_n} con una *cross-entropy loss* per forzare un allineamento tenendo conto anche delle etichette. Per quanto riguarda gli archi è possibile sostituire Q_{T_n} con $Q_{T_e} = \Pr(y_j|u_i)$, cioè la distribuzione di probabilità delle etichette per gli archi collegati a u_i: ad esempio se u_i ha 5 archi, 2 dei quali sono rossi e gli altri 3 gialli, $Q_{T_e}(rosso|u_i) = 0.4$. Con un procedimento simile è possibile applicare la stessa penalizzazione al secondo livello di *attention*.

La funzione di *loss* così ottenuta viene moltiplicata per un coefficiente C_a (iperparametro) e sommata a quella del classificatore multilabel.

Divergenza tra grafi

A questo punto abbiamo un autoencoder H_S in grado di riprodurre la struttura di un prototipo G_S, e un encoder "aumentato" H_T con due livelli di *attention* addestrato a riprodurre la struttura di G_T **usando** quella di G_S. Se i due grafi sono molto simili, ci aspettiamo che H_T ottenga dei risultati molto

buoni; al contrario se i grafi sono diversi H_T non sarà in grado di catturare correttamente la struttura di G_T.

È possibile quindi definire una misura di divergenza di un grafo G_T da un prototipo G_S come

$$\mathcal{D}'(G_T||G_S) = \sum_{v_i \in G_T} \sum_{j|e_{ji} \in G_T} -\log \Pr(v_j|v_i, H_S). \tag{3.9}$$

Dato che H_S non è in grado di riprodurre perfettamente la struttura di G_S si può assumere che $\mathcal{D}'(G_S||G_S) \neq 0$: perciò si definisce

$$\mathcal{D}(G_T||G_S) = \mathcal{D}'(G_T||G_S) - \mathcal{D}'(G_S||G_S) \tag{3.10}$$

in modo che $\mathcal{D}(G_S||G_S) = 0$. Da notare che $\mathcal{D}(G_T||G_S)$ potrebbe differire da $\mathcal{D}(G_S||G_T)$: se è necessaria la simmetria è possibile utilizzare $\mathcal{D}(G_S, G_T) = \mathcal{D}(G_S||G_T) + \mathcal{D}(G_T||G_S)$.

Embedding di grafi

Infine si usano i valori di divergenza per rappresentare i grafi in un *dissimilarity space*. Considerando un insieme di N prototipi, la funzione di *embedding* è definita come

$$\Psi(G_T) = [\mathcal{D}(G_T, G_{S_0}), \mathcal{D}(G_T, G_{S_1}), \mathcal{D}(G_T, G_{S_2}), \ldots, \mathcal{D}(G_T, G_{S_N})]. \tag{3.11}$$

3.5.2 Considerazioni

Questo metodo è stato considerato per l'aggregazione di eventi ottenuti con DeepEventMine. Risolve le principali problematiche descritte per SimGNN (Sezione 3.4.2):

- non è supervisionato;
- calcola degli *embedding* che possono quindi essere usati anche per il *clustering* e per valutare facilmente la similarità fra qualsiasi coppia;
- tiene conto sia degli attributi dei nodi che di quelli degli archi;
- si ottiene una mappatura di equivalenza tra i nodi dei due grafi.

La mappatura tra i nodi viene calcolata solamente rispetto ai prototipi; è possibile però applicare lo stesso metodo tra coppie arbitrarie di grafi considerate particolarmente interessanti.

Capitolo 4

Contributi

In questo capitolo viene descritto il lavoro svolto, gli esperimenti effettuati e i risultati ottenuti.

Inizialmente viene definita la struttura della soluzione proposta per raggiungere gli obiettivi, che successivamente viene realizzata concretizzando i passaggi richiesti con le implementazioni analizzate nel Capitolo 3. Tutto il codice presentato è scritto in linguaggio Python, usando come *framework* PyTorch[1] o TensorFlow[2].

4.1 Struttura proposta

Figura 4.1: Struttura della soluzione proposta

[1] `https://pytorch.org/`
[2] `https://www.tensorflow.org/`

La struttura identificata per il raggiungimento degli obiettivi definiti nella Sezione 2.3 è illustrata in Figura 4.1 ed è composta dai seguenti macro-blocchi.

- **Pre-elaborazione** Il testo viene preparato per essere successivamente analizzato, in base alle necessità della specifica implementazione.
- **Estrazione delle entità** Le entità contenute nel testo vengono riconosciute e classificate.
- **Estrazione di relazioni o eventi** Vengono estratte le interazioni espresse tra le entità identificate; a seconda dell'implementazione utilizzata vengono rappresentate come *relazioni* o *eventi* (vedi Sezione 1.2).
- **Aggregazione** Si aggregano gli eventi / relazioni estratti/e sulla base del loro contenuto semantico consentendo la quantificazione dei fatti menzionati nel *corpus* di riferimento. Se necessario vengono aggregate anche le entità, ancora una volta in base al contenuto semantico.

Di seguito vengono presentate specifiche istanze di questa struttura, utilizzando le soluzioni presentate nel Capitolo 3 come implementazioni dei diversi macro-blocchi.

4.2 Language Models are Open Knowledge Graphs

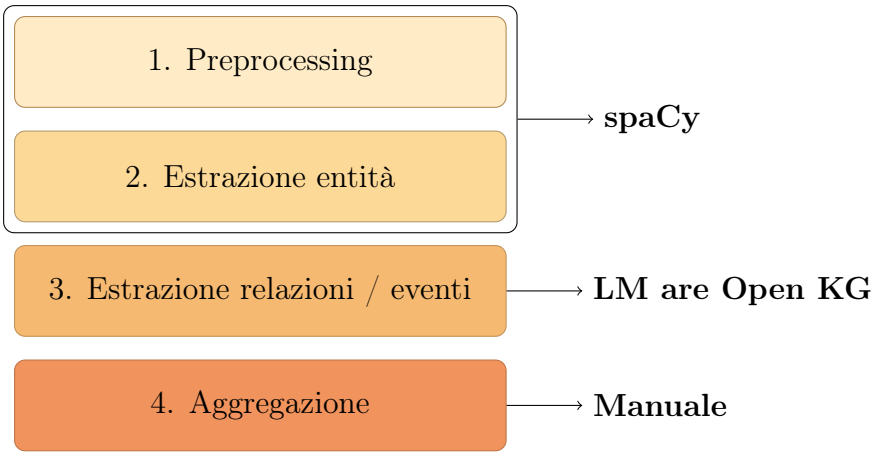


Figura 4.2: Struttura della soluzione usando *Language Models are Open Knowledge Graphs*.

In questo caso viene usato il metodo presentato in *Language Models are Open Knowledge Graphs* nella Sezione 3.1 per l'estrazione di relazioni. Sia la pre-elaborazione che il riconoscimento delle entità vengono effettuate usando la libreria **spaCy** descritta in precedenza. L'aggregazione viene fatta con un metodo manuale sfruttando il calcolo della similarità semantica (Sezione 1.5).

4.2.1 Implementazione di riferimento

Gli autori non forniscono una implementazione, per cui ne è stata utilizzata una non ufficiale pubblicata su GitHub[3], basata su framework PyTorch. Da un'analisi del codice sono emerse molte discrepanze con il procedimento descritto nel paper, oltre che un'implementazione non ottimale che ne limitava di molto l'efficienza.

- **Attention per modelli con decoder** Come indicato in Sezione 3.1.1, per passare da un parola all'altra viene usata l'*attention* della prima verso la seconda. Purtroppo però i modelli basati su *decoder* (come GPT-2) usano la *Masked Self-Attention* in cui sono presenti valori solamente verso i *token* precedenti (vedi Sezione 1.1). Per fare in modo che funzionino correttamente ho considerato l'*attention* della seconda parola verso la prima (nell'esempio di Figura 3.2, l'attention di *is* verso *Dylan* piuttosto che viceversa).

- **Nessuna risoluzione delle coreferenze** Nella Sezione A.1 di [15] gli autori indicano di aver utilizzato la libreria `neuralcoref`[4] per sostituire i pronomi con il nome a cui facevano riferimento. In questa implementazione non veniva eseguito per cui molti fatti candidati contenevano pronomi nella *testa* o nella *coda*; per risolvere ho modificato il testo originale sostituendo i pronomi tramite *neuralcoref*.

- **Utilizzo dell'*attention* del primo livello** Gli autori indicano chiaramente di aver ottenuto risultati migliori utilizzando i valori di *attention* dell'ultimo livello del modello; per un errore di indicizzazione, questa implementazione usava invece i valori ottenuti dal primo livello. Ho quindi provveduto a modificare il codice perché rispecchiasse le indicazioni degli autori.

- **Filtri sui fatti** I fatti candidati vengono filtrati secondo tre criteri: il valore di confidenza, la frequenza della relazione e il fatto che la relazione sia contigua. Solamente il primo era applicato nel codice; io ho aggiunto solo il terzo, perché il secondo sarebbe stato problematico.

[3] `https://github.com/theblackcat102/language-models-are-knowledge-graphs-pytorch`
[4] `https://github.com/huggingface/neuralcoref`

```
{"line": 0, "tri": [
  {"h": "Prof_Costamagna", "r": "work", "t": "Gemelli", "c": 0.13}
]}

{"index": 0,
"text": "Prof Costamagna works at Gemelli. Prof Costamagna
  performs the POEM surgery",
"tri": [
  {"h": [0, 2], "r": [2, 3], "t": [4, 5], "c": 0.03, "sentence_index": 0},
  {"h": [6, 8], "r": [8, 9], "t": [9, 12], "c": 0.04, "sentence_index": 1}
]}
```


Figura 4.3: Un esempio di triple prima e dopo la modifica del codice, estratte dal testo *"Prof Costamagna works at Gemelli. He performs the POEM surgery"*.

- **Risultati scadenti nella fase di Map** Gli autori non indicano lo strumento specifico utilizzato per la fase di *Map*; in questo caso veniva utilizzata la libreria `REL`[5], che usa Wikidata come *Knowledge Base* di riferimento. Purtroppo però molto spesso il testo veniva collegato a entità con cui non aveva nulla a che fare, ma con cui magari condivideva una piccola porzione di testo. Sospetto che il motivo sia la presenza nel nostro testo di terminologia specifica, non correttamente rappresentata all'interno di Wikipedia, e che spesso contiene errori di battitura che possono aver contribuito alla bassa qualità di questi risultati. Per evitare di perdere il significato originale dei termini ho quindi preferito rimuovere completamente questo passaggio.
- **Scarsa efficienza** L'elaborazione era piuttosto lenta inizialmente. In primo luogo perché alcune operazioni venivano parallelizzate senza che ce ne fosse la necessità: l'*overhead* necessario per creare i processi e suddividere il lavoro era molto maggiore del vantaggio che se ne traeva. Rimuovendo questa parallelizzazione è diventato circa **quattro volte più veloce**. In secondo luogo ho modificato il codice in modo da sfruttare l'inferenza in *batch*: analizzando più frasi contemporaneamente sono riuscito a velocizzare l'elaborazione ulteriormente di circa **tre volte** (un totale di dodici volte rispetto all'inizio). Questo ha reso estremamente più facile e veloce effettuare esperimenti sul codice.

Inoltre ho modificato il codice in modo da mantenere molte informazioni nelle triple risultanti. Piuttosto che rappresentare *testa*, *relazione* e *coda* come

[5] `https://github.com/informagi/REL/`

		prevent	cappuccino
pain	2 pain	prevents	romagna
pains	my pains	prevented	2005
a "pain	pain etc	preventive	soccer
no pain	pain.then	preventing	either

(a) Entità `pain` (b) Relazione `prevent` (c) Relazioni insensate

Figura 4.4: Alcuni esempi di entità e relazioni estratte dal testo

semplice testo, ho salvato gli intervalli di parole all'interno del testo originale. In questo modo è possibile effettuare ulteriori elaborazioni avendo a disposizione tutte le informazioni fornite da `spaCy` sulle parole, come *Part Of Speech*, albero di dipendenze etc.

In Figura 4.3 si vede un esempio di triple estratte rispettivamente prima e dopo le modifiche, dal testo *"Prof Costamagna works at Gemelli. He performs the POEM surgery"*.

Nel primo caso i componenti dell'unica tripla estratta vengono rappresentati come semplice testo. La seconda frase viene ignorata perché il soggetto è un pronome. Si può notare anche che gli spazi vengono sostituiti da un trattino basso (*Prof_ Costamagna*) e che i verbi vengono lemmatizzati (*work*).

Nel secondo caso invece viene mantenuto il testo originale pre-elaborato in cui sono state risolte le coreferenze (si vede che *"He"* è diventato *"Prof Costamagna"*), e le triple sono rappresentate da intervalli di parole all'interno del testo. In questo caso corrispondono a Viene anche mantenuta l'informazione (ridondante ma comoda) su quale frase contiene la tripla. Ovviamente in questo caso non vengono effettuate le operazioni sul testo come in precedenza (ad esempio lemmatizzazione), ma possono essere facilmente effettuate in un secondo momento: non viene persa nessuna informazione sul documento originale e questo permette una maggiore flessibilità.

Il codice modificato e incrementato è pubblicato su GitHub[6].

4.2.2 Aggregazione

Avendo rimosso il passaggio di **Map**, le entità e relazioni risultanti non sono altro che porzioni estratte dal testo e non corrispondono a uno schema particolare. Alcuni esempi di entità sono rappresentate in Figura 4.4a: il concetto principale (`pain`) è spesso affiancato da punteggiatura, articoli, aggettivi o altri termini[7]. Per le relazioni sono presenti varie declinazioni dello stesso verbo o

[6] `https://github.com/Carlovan/language-models-are-knowledge-graphs-pytorch/`

[7] Da notare che queste "frasi" sono state identificate come gruppi nominali

espressione (Figura 4.4b), oltre che una grande quantità di termini poco adatti a rappresentare una relazione (Figura 4.4c).

Il primo obiettivo è stato quello di cercare di **aggregare** inizialmente le entità che facessero riferimento allo stesso concetto (ad esempio *"the pain"* e *"my pains"*). Si è scelto di basarsi sulla **similarità semantica** (vedi Sezione 1.5) per determinare quali "frasi" avessero lo stesso significato; in particolare usando il modello `stsb-roberta-large`.

L'aggregazione di relazioni non è stata effettuata a causa degli scarsi risultati ottenuti.

Aggregazione di entità

Si è utilizzato un approccio "gerarchico" per aggregare tra di loro le entità a vari livelli di confidenza. Nell'esempio di Figura 4.4a, si può considerare che *"the pain"* è esattamente equivalente a *"pain"*, mentre *"no pain"* è sempre relativo allo stesso concetto ma semanticamente si trova lontano da esso.

Inizialmente sono state create delle macro categorie basandosi solamente sul sostantivo principale dei gruppi nominali, per poter ridurre la quantità di coppie da confrontare entro un valore gestibile. All'interno di ogni macro categoria i gruppi nominali vengono ordinati per numero di termini crescente, e viene misurata la similarità semantica di ogni gruppo con tutti i successivi. In questo modo viene usata una semplice euristica per determinare quanto un'espressione sia generale: più termini avrà e più sarà specifica. Possiamo quindi dire che se due espressioni sono semanticamente simili, quella più specifica è "inclusa" in quella più generica (ad esempio, *"pain"* è più generico di *"retrosternal pain"*). Il risultato di questa operazione è un grafo diretto aciclico in cui il peso degli archi è il valore di similarità tra due espressioni; su di esso è possibile calcolare il massimo albero coprente per mantenere le coppie con maggiore similarità semantica.

```
# Sostantivo principale del gruppo nominale
root = ...
# Lista di noun chunks che hanno 'root' come sostantivo principale
mentions = ...
# Modello utilizzato per la similarita' semantica
semsim_model = ...
to_merge = sorted(set(mentions), key=lambda e: len(e.split()))
text_pairs = list(it.combinations(to_merge, 2))
scores = semsim_model.predict(text_pairs)

similarity_graph = networkx.Graph()
for (ent_a, ent_b), score in zip(text_pairs, scores):
```

```
    similarity_graph.add_edge(ent_a, ent_b, weight=score)

tree = networkx.maximum_spanning_tree(similarity_graph)
```

Listato 4.1: Codice per l'aggregazione gerarchica delle entità

4.2.3 Risultati ottenuti

A causa della scarsissima qualità dei fatti estratti dal metodo in esame, l'aggregazione manuale utilizzando similarità semantica risulta estremamente complessa da gestire e realizzare correttamente, ottenendo comunque risultati scadenti. Per questo si è deciso di proseguire la ricerca sperimentando soluzioni differenti.

4.3 ATLOP

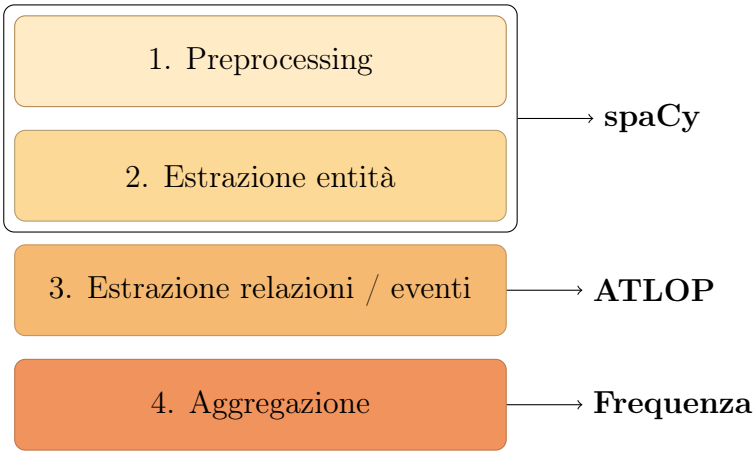


Figura 4.5: Struttura della soluzione usando ATLOP.

In questo caso viene utilizzato il modello ATLOP descritto nella Sezione 3.2 per l'estrazione di relazioni. Sia la pre-elaborazione che il riconoscimento di entità vengono effettuati utilizzando la libreria **spaCy** descritta in precedenza. L'aggregazione viene effettuata semplicemente sulla base delle frequenza con cui le relazioni vengono espresse nel testo.

4.3.1 Implementazione di riferimento

L'implementazione ufficiale fornita dagli autori (disponibile su GitHub[8]) contiene sia il codice per l'inferenza sia quello per l'addestramento, che utilizza il framework PyTorch. Sono inoltre disponibili i due modelli preaddestrati valutati nel paper, basati su *language model* BERT-BASE e RoBERTa-LARGE. Il dataset utilizzato è DocRED [85]: contiene circa 5000 documenti, presi da Wikipedia, in cui sono state annotate le relazioni presenti a livello di documento. Eseguendo il modello fornito si ottengono gli stessi risultati indicati nel paper.

```
def find_entities(entity_name: str, doc: spacy.tokens.Doc):
    ents: List[dict] = []
    name_re = re.compile(fr'(\b|_){entity_name}(\b|_)')
    for sent_i, sent in enumerate(doc.sents):
        for nc in sent.noun_chunks:
            if name_re.search(nc.text) is not None:
                ents.append({
                    'pos':(nc.start - sent.start, nc.end - sent.start),
                    'sent_id': sent_i,
                    'name': nc.text
                })
    return ents
```

Listato 4.2: Codice per identificare le menzioni di una entità usando **spaCy**

Il codice si aspetta che i dati abbiano lo stesso formato utilizzato in DocRED:

- i documenti sono suddivisi in frasi e la frasi in parole;
- le menzioni sono identificate all'interno del testo;
- per ogni entità si sanno le menzioni che la riguardano.

Per poterlo testare su dati personalizzati ho realizzato un semplice script (`text2docred.py`) che converte un qualsiasi testo nel formato sopra descritto, con un algoritmo molto semplice e utilizzando la libreria **spaCy**. La suddivisione in frasi e parole viene effettuata automaticamente. Per le menzioni ho utilizzato direttamente i *noun chunks*, mantenendo solamente quelli che contengono determinate parole di interesse, definite manualmente, ognuna delle quali identifica una particolare entità. Le menzioni così ottenute associate all'entità corrispondente. Per verificare se un *noun chunk* contiene una specifica parola viene utilizzata l'espressione regolare `(\b|_)PAROLA(\b|_)` in modo da catturare solamente quella specifico termine: ad esempio se si è interessati a *"pain"*

[8] `https://github.com/wzhouad/ATLOP`

bisogna evitare di selezionare anche i documenti contenenti *"Spain"*. Nell'espressione regolare oltre al carattere di *word boundary* (`\b`) viene considerato anche il trattino basso perché nel dataset alcuni termini di particolare interesse sono stati aggregati con esso (ad esempio `gastroesophageal_reflux_disease`). Infine tutti i documenti che contengono menzioni relative a meno di due entità vengono ignorati, dato che non possono esprimere alcuna relazione.

Il risultato del modello rappresenta le relazioni utilizzando gli identificatori delle entità e dei predicati utilizzati nel dataset di input. Un secondo script (`result2text.py`) sostituisce gli identificatori nel risultato con i contenuti testuali corrispondenti, per permettere una migliore valutazione dei risultati.

Il codice che ho sviluppato è pubblicato su GitHub[9].

4.3.2 Addestramento

Il dataset DocRED [85] è stato creato partendo da circa 5000 pagine di Wikipedia e contiene solamente 96 tipologie di relazioni, tutte utilizzabili in ambiti molto generali: aziende, Paesi, personaggi famosi, entità geografiche... Quasi nessuna è utile al nostro caso di studio, in cui invece si parla principalmente di un contesto quotidiano, in un dominio specifico.

Per questo ho riaddestrato un modello sfruttando anche il dataset FewRel [86] che contempla 743 relazioni, di cui 86 presenti anche in DocRED. FewRel però contiene solamente relazioni a livello di frase, un limite che non permetterebbe di utilizzare appieno le potenzialità di ATLOP. Ho costruito quindi un nuovo dataset unendo DocRED e FewRel, ottenendone uno con 753 tipologie di relazioni, con lo scopo di ottenere un modello in grado di riconoscere relazioni a livello di documento e di gestire tutte le tipologie definite in FewRel.

L'addestramento è stato effettuato su Google Colab[10] utilizzando il modello basato su BERT-BASE e ha richiesto circa 10 ore.

4.3.3 Risultati ottenuti

Per effettuare gli esperimenti il testo è stato pre-elaborato utilizzando **spaCy** come descritto in precedenza. In particolare sono state estratte dal dataset originale (Sezione 2.2.3) le entità *"costamagna"*, *"gemelli"*, *"poem"* e *"reflux"*: il Professor Costamagna e il suo team di ricercatori lavorano all'ospedale Gemelli di Roma e sono tra i pochi in Italia ad effettuare l'operazione di tipo POEM, un intervento chirurgico con lo scopo di ridurre i sintomi dell'acalasia esofagea, tra cui il reflusso gastrico. Il dataset ottenuto è formato da 371 documenti (post) per un totale di 768 menzioni.

[9] `https://github.com/Carlovan/ATLOP`

[10] `https://research.google.com/colaboratory/faq.html`

Applicando i modelli preaddestrati su questi dati non viene riconosciuta alcuna relazione. Dopo averli riaddestrati sul dataset costruito combinando DocRED e FewRel i risultati sono leggermente migliorati: vengono identificate 100 triple in 91 documenti, che utilizzano 13 diverse classi di relazioni. Di seguito sono indicate alcune delle 33 triple distinte estratte, accompagnate dal numero di occorrenze, in cui sono state evidenziate quelle più interessanti.

poem → characters → reflux [12]
poem → characters → costamagna [11]
costamagna → notable work → poem [9]
poem → location → gemelli [7]
poem → location → costamagna [6]
costamagna → field of work → poem [6]
poem → publisher → gemelli [5]
poem → publisher → costamagna [5]
gemelli → headquarters location → costamagna [4]
costamagna → instance of → poem [4]
poem → said to be the same as → reflux [3]
poem → characters → gemelli [2]
gemelli → subsidiary → costamagna [2]
gemelli → instance of → poem [2]
costamagna → subsidiary → gemelli [2]
costamagna → said to be the same as → poem [2]
reflux → instance of → poem [1]
reflux → characters → gemelli [1]
poem → said to be the same as → gemelli [1]
poem → notable work → gemelli [1]
poem → instance of → costamagna [1]
poem → headquarters location → costamagna [1]
poem → author → gemelli [1]
gemelli → said to be the same as → costamagna [1]
gemelli → owned by → costamagna [1]
gemelli → notable work → poem [1]
costamagna → said to be the same as → gemelli [1]
costamagna → publisher → gemelli [1]
costamagna → location → gemelli [1]
costamagna → has part → poem [1]

La relazione *characters*[11] significa che l'oggetto è un personaggio che fa parte del soggetto (il quale dovrebbe essere un'opera come un film, uno spettacolo,

[11] https://www.wikidata.org/wiki/Property:P674

un videogioco...): in questo caso non è applicabile con questa semantica, ma potrebbe essere interpretato con il significato di avere un ruolo rispetto al soggetto, che è in effetti corretto. Oltre a poche eccezioni le relazioni estratte sono perlopiù insensate (`gemelli` → `instance of` → `poem`), invertite (`gemelli` → `headquarters location` → `costamagna`) o contrarie alla realtà (`poem` → `said to be the same as` → `reflux`).

In definitiva i risultati sono davvero scadenti. In primo luogo le tipologie di relazioni non rispecchiano perfettamente quelle espresse nei dati in esame: un esempio è la relazione *characters*, la cui semantica deve essere reinterpretata per applicarla al caso di interesse. Anche per questo viene identificato solo un sottoinsieme delle relazioni esistenti nel testo (ad esempio non ve ne è nessuna tra *"reflux"* e *"poem"*), e molte di quelle estratte sono in realtà scorrette o insensate. Per questi motivi ci è mossi verso altre soluzioni più efficaci.

4.4 Soluzione definitiva: DeepEventMine e DDGK

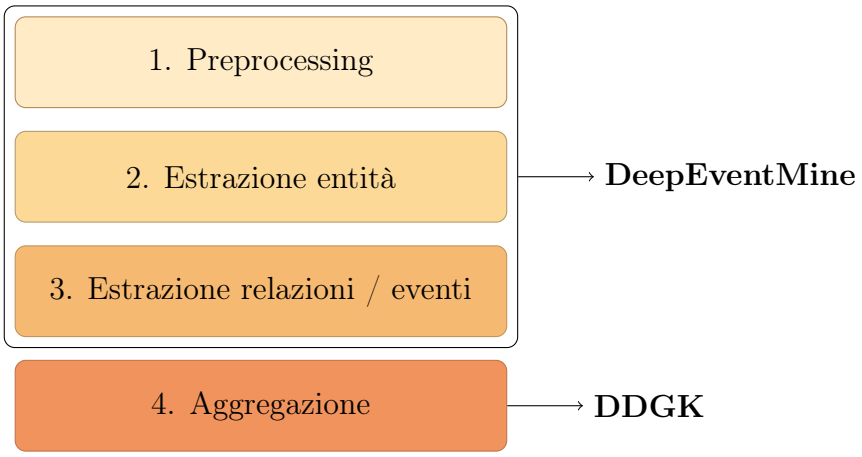


Figura 4.6: Struttura della soluzione usando DeepEventMine e DDGK.

In questo caso viene usato DeepEventMine sia per il preprocessing che per l'estrazione di entità ed eventi. Questi ultimi vengono quindi rappresentati come grafi di cui vengono calcolati degli *embedding* sfruttando DDGK, che permettono poi sia di effettuare *clustering* che di calcolare velocemente la similarità tra qualsiasi coppia. I risultati sono i migliori tra quelli ottenuti fin'ora e sono molto promettenti.

4.4.1 Implementazione DeepEventMine

L'implementazione ufficiale pubblicata dagli autori (disponibile su GitHub[12]) utilizza il framework PyTorch e contiene solamente il codice per l'inferenza, mentre quello per l'addestramento non è stato rilasciato "a causa di alcuni requisiti del progetto"[13]. Sono presenti però molti script utili che permettono di scaricare sia i **sette** dataset utilizzati per l'addestramento sia i modelli preaddestrati, oltre che fornire una comoda interfaccia per l'inferenza anche di dati completamente nuovi.

Dato che vengono estratte contemporaneamente entità e relazioni, l'input è formato da semplici documenti testuali. Gli script forniti si occupano della pre-elaborazione, suddividendo i documenti in frasi e in parole, e costruendo una mappatura a livello di carattere tra il testo originale e quello suddiviso: in questo modo sarà possibile ricostruire i riferimenti al testo originale a partire dai risultati finali.

Viene generato un file per ogni documento contenente le annotazioni di entità ed eventi individuati, salvato in formato *"standoff"* (ufficialmente utilizzato per il workshop di "BioNLP Shared Tasks"[14]), ad esempio:

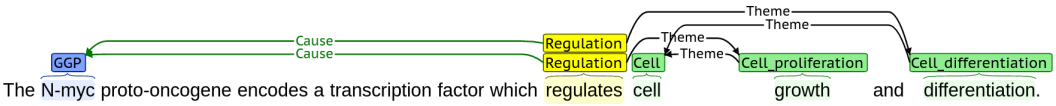
```
T1      Protein 0 7     RFLAT-1
T2      Positive_regulation 53 62       activates
E1      Positive_regulation:T2 Theme:E2 Cause:T1
M1      Negation E1
```

Il primo valore è un identificatore univoco che indica anche il tipo di costrutto: `T` significa entità, `E` significa evento e `M` significa modificatore (ad esempio "negazione", "speculazione"...). Per le entità è indicata la classe e l'intervallo della menzione nel testo; per gli eventi viene indicato il *trigger* e la lista degli argomenti (che possono essere entità o eventi) con i propri ruoli; per i modificatori viene indicato il tipo e l'evento a cui si riferiscono. Lo stesso formato viene utilizzato da uno strumento di visualizzazione chiamato Brat[15] che permette di valutare facilmente i risultati ottenuti.

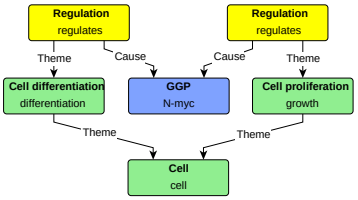
DeepEventMine è in grado di estrarre eventi innestati a più livelli che possono essere rappresentati come grafi diretti aciclici. Il grafo in Figura 4.7b può essere suddiviso in due diversi eventi parzialmente sovrapposti ma che possiedono semantiche differenti, radicati rispettivamente nei due nodi "regulates": le entità in comune sono solamente *"N-myc"* e *"cell"*.

Per costruire questa rappresentazione ho realizzato uno script (`brat2graphs.py`) che converte le annotazioni in formato Standoff in grafi diretti aciclici. La

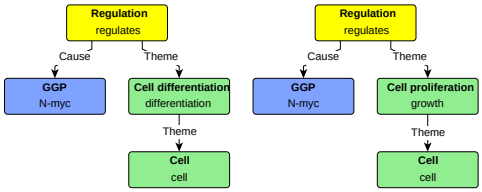
[12] `https://github.com/aistairc/DeepEventMine`
[13] `https://github.com/aistairc/DeepEventMine/issues/3`
[14] `http://2011.bionlp-st.org/home/file-formats`
[15] `https://brat.nlplab.org/`



(a) Il testo con gli eventi identificati



(b) Il grafo diretto aciclico corrispondente



(c) I due eventi separati, parzialmente sovrapposti

Figura 4.7: Un esempio di evento innestato preso dal dataset *Cancer Genetics*

manipolazione dei grafi è stata gestita utilizzato una libreria dedicata chiamata `networkx`[16]. Per ogni documento viene inizialmente costruito un grafo direzionato contenente tutte le entità, in cui ogni *trigger* è collegato ai suoi argomenti con archi a cui è associato il ruolo corrispondente; nel caso di eventi innestati, il *trigger* dell'evento padre è collegato a quello dell'evento figlio. Chiaramente si tratterà di un grafo non connesso, perché gli eventi sono quasi sempre disgiunti. A questo punto vengono identificate come radici degli eventi quei nodi che non possiedono archi in ingresso; i nodi completamente isolati (cioè che non possiedono archi né in ingresso né in uscita) vengono ignorati, dato che non fanno parte di alcun evento. Infine si ottengono tutti i nodi raggiungibili dalle radici per costruire un grafo diretto aciclico per ogni evento (che possono essere parzialmente sovrapposti come nell'esempio precedente). Tutti i grafi vengono salvati in un file in formato JSON per essere successivamente utilizzati.

Durante l'elaborazione del testo da parte di DeepEventMine vengono generati degli *embedding* per le entità identificate, che possono essere estremamente utili nella fase di aggregazione per rappresentare il loro significato semantico. Normalmente l'elaborazione termina una volta ottenute le annotazioni in formato Standoff per cui vengono scartati. Perciò ho modificato il codice in modo da salvare su file gli *embedding* in modo da poter essere successivamente utilizzati. Ho incontrato alcuni problemi a causa del grande utilizzo di memoria richiesto, per cui è stato necessario salvare i dati intermedi su file temporanei piuttosto che mantenerli in memoria centrale fino alla fine.

Le modifiche che ho apportato al codice, compreso lo script descritto sopra, sono disponibili su GitHub[17].

[16] `https://networkx.org/`

[17] `https://github.com/Carlovan/DeepEventMine`

4.4.2 Implementazione DDGK

Il codice dell'algoritmo, realizzato con il framework TensorFlow, è pubblicato nel repository ufficiale di *google-research*[18] insieme a moltissime altre implementazioni. Nel metodo descritto gli *encoder* e i livelli di *attention* vengono addestrati direttamente sui grafi in esame e quindi non è possibile creare un modello preaddestrato. Il codice è in linea con il contenuto del *paper* ed eseguendolo si ottengono risultati solo leggermente più bassi quelli indicati nel paper per il dataset MUTAG.

La prima modifica è stata effettuata per poter usare il codice anche con Tensorflow 2.x in modo da poter sfruttare la versione preinstallata su Google Colab e ottimizzata per l'utilizzo della GPU. In TensorFlow2 è incluso un layer di compatibilità che permette di usare quasi tutte le funzionalità di TensorFlow1. Alcuni moduli secondari però sono stati rimossi e riorganizzati in librerie esterne: non esiste un chiara documentazione su come effettuare la migrazione, per cui mi sono dovuto basare su varie risorse informali come forum o *Issue* pubblicate su GitHub. La classe `HParams` utilizzata per gestire gli iperparametri è stata invece completamente rimossa: l'ho sostituita con una semplice classe contenente i valori degli iperparametri in appositi attributi.

In secondo luogo ho modificato lo script principale (`main`) per poter funzionare nel nostro caso specifico. Fortunatamente i grafi venivano già rappresentati e gestiti usando `networkx` per cui ho potuto utilizzare direttamente quelli ottenuti dagli eventi come descritto nella sezione precedente. Ho rimosso il codice relativo alla valutazione del dataset MUTAG sostituendolo con il salvataggio su file degli *embedding* calcolati, in formato JSON, in modo da poterli usare successivamente.

Infine è stato necessario effettuare alcune modifiche al modello: il codice originale considerava che ogni nodo avesse un'etichetta discreta (vedi Sezione 3.5.1) mentre nel nostro caso possiedono dei vettori numerici (gli *embedding* delle entità calcolati da DeepEventMine). Ho modificato la funzione di *loss* per la coerenza degli attributi in modo da calcolare il valore atteso degli *embedding* dei nodi del grafo G_S sulla base delle predizione del primo livello di *attention* $\mathcal{M}_{T \to S}$, usando poi la **distanza coseno** per confrontare il risultato con l'*embedding* del nodo originale di G_T.

```
def NodesEmbeddings(graph):
  # embeddings size is (graph_num_node, D)
  embeddings = [graph.node[n]['embedding'] for n in graph.nodes()]
  assert len(set(map(len, embeddings))) == 1, 'Some embeddings have
      different size'
```

[18] https://github.com/google-research/google-research/tree/master/graph_embedding/ddgk

```
  return tf.convert_to_tensor(embeddings, dtype=tf.float64)

def NodeEmbeddingLoss(source, source_node_prob, target):
  # source_embeddings size is (source_num_node, D) and is normalized
  source_emb = tf.linalg.l2_normalize(NodesEmbeddings(source), axis=1)

  # target_embeddings size is (target_num_node, D) and is normalized
  target_emb = tf.linalg.l2_normalize(NodesEmbeddings(target), axis=1)

  predicted_emb = tf.matmul(source_node_prob, source_emb)

  # The reduction is actually a mean
  return tf.cast(tf.losses.cosine_distance(predicted_embeddings,
      target_embeddings, axis=1,
      reduction=tf.losses.Reduction.SUM_OVER_BATCH_SIZE), tf.float64)
```

Listato 4.3: *Loss* per gestire gli *embedding* dei nodi

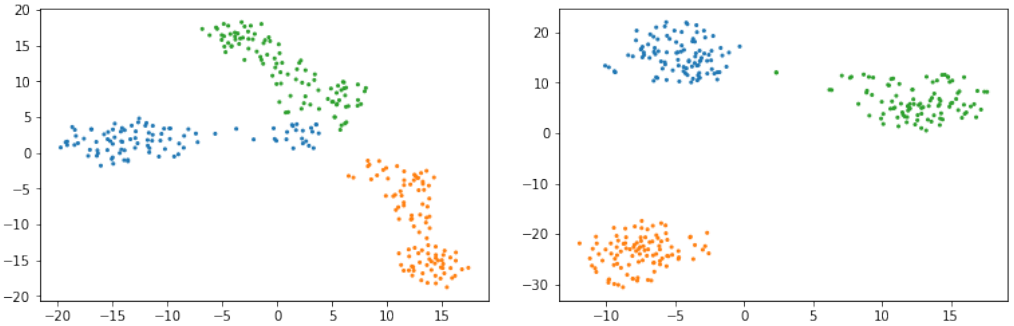
Il codice con le modifiche effettuate è pubblicato su GitHub[19].

4.4.3 Risultati ottenuti

Come detto nella Sezione 3.3.2 il codice per l'addestramento di DeepEventMine non è disponibile, per cui sono stati utilizzati solamente i sette modelli preaddestrati. La qualità dei risultati ottenuti è ottima applicando ogni modello sul dataset su cui è stato addestrato, ma cala sensibilmente allontanandosi anche di poco da quello specifico dominio. Utilizzandoli sui nostri dati non viene riconosciuto alcun evento, e anche analizzando un insieme di documenti presi da PubMed aventi come tema l'acalasia esofagea vengono estratti non più di un paio di eventi per documento, tutti poco significativi perché di ambito molto generico. Per questo si è deciso di utilizzare solamente i dataset ufficiali con i modelli corrispondenti.

In particolare sono stati estratti gli eventi contenuti in **tre** di essi: "Cancer Genetics"[20] (CG), "Epigenetics and Post-translational Modifications"[21] (EPI) e "Infectious Diseases"[22] (ID). Sono stati quindi costruiti i grafi associati: 7714 per CG, 2464 per EPI e 1697 per ID. Per ogni dataset sono stati selezionati casualmente 100 grafi, in modo da limitare il tempo necessario alla successiva elaborazione e ottenere un dataset bilanciato. Usando DDGK si sono ottenuti gli *embedding* a 16 dimensioni, calcolati con un insieme di 16 prototipi scelti

[19] https://github.com/Carlovan/google-research/
[20] http://2013.bionlp-st.org/tasks/cancer-genetics
[21] http://2011.bionlp-st.org/home/epigenetics-and-post-translational-modifications
[22] http://2011.bionlp-st.org/home/infectious-diseases



(a) Vettori ottenuti da DDGK.

(b) Versione normalizzata dei vettori ottenuti da DDGK.

Figura 4.8: *Embedding* dei grafi ottenuti con DDGK, visualizzati con t-SNE e colori in base al dataset di appartenenza.

in modo casuale. Il coefficiente C_a della funzione di *loss* per la coerenza degli attributi è stato impostato a 5. In Figura 4.8 è riportata la proiezione in due dimensioni dei risultati calcolata con T-SNE [87] (con *perplexity* = 30): il colore dei punti rappresenta il dataset di appartenenza dell'evento corrispondente. In Figura 4.8b sono visualizzati gli stessi vettori, ma normalizzati in modo da essere unitari.

Per determinare la qualità dei risultati vengono usati due principali criteri di valutazione.

- La **separabilità lineare**, cioè se è possibile determinare degli iperpiani che dividano lo spazio in regioni in modo che tutti i punti appartenenti allo stesso dataset si trovino nella stessa regione, e che punti appartenenti a dataset differenti di trovino in regioni differenti. Se questo fosse vero significherebbe che è possibile determinare il dataset a cui appartiene un evento solo sulla base del suo *embedding*, il quale ne deve quindi codificare la caratteristiche peculiari.

- La **presenza di *cluster*** ben definiti e individuabili automaticamente. Se tali *cluster* rispecchiassero l'appartenenza ai dataset significherebbe che gli eventi dallo stesso dataset sono più vicini tra di loro rispetto che a eventi di altri dataset, e quindi le similarità semantiche sono in qualche modo rispettate.

Infine viene fatta una semplice valutazione qualitativa su alcune coppie di eventi teoricamente molto simili.

Separabilità lineare

Per determinare la **separabilità lineare** è stato addestrato un semplice modello di regressione logistica sui vettori a 16 dimensioni ottenuti da DDGK, usando il dataset di appartenenza come classe. Il modello è in grado di separare perfettamente i dati anche con una forte regolarizzazione (iperparametro $C = 0.02$).

```
embeddings = ... # Vettori con 16 dimensioni ottenuti da DDGK
graph_labels = ... # Dataset di appartenenza per ogni embedding
log_clf = LogisticRegression(C=0.02, random_state=42)
    .fit(embeddings, graph_labels)
print("Logistic regression score =",
    log_clf.score(embeddings, graph_labels))
```

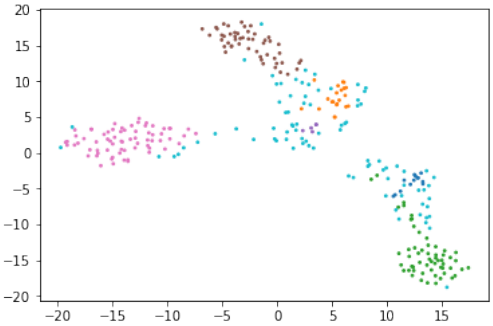
Listato 4.4: Modello di regressione logistica per la separazione lineare degli eventi.

Clustering

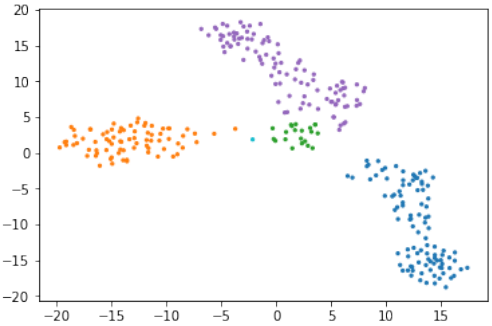
Per determinare la presenza di ***cluster*** invece è stato usato l'algoritmo HDBSCAN [88], il cui risultato viene valutato utilizzando *V-Measure* [89] che misura sia se tutti i vettori con la stessa classe sono nello stesso *cluster* (*completeness*) sia se tutti gli elementi di un *cluster* sono della stessa classe (*homogeneity*).

HDBSCAN è stato applicato sia sui vettori a 16 dimensioni ottenuti da DDGK, sia sulla loro proiezione in 2 dimensioni calcolata con T-SNE. Nel secondo caso vengono messe in evidenza le dimensioni più significative dello spazio, cercando di preservare le similarità tra i vettori originali. In entrambi i casi viene eseguita una ricerca esaustiva per determinare i parametri ottimali.

In Figura 4.9 sono visibili i risultati.
Sulla sinistra (con vettori a 16 dimensioni) vengono individuati 7 cluster diversi, che non rispecchiano l'appartenenza ai dataset. Il punteggio *V-Measure* è 0.563 e i valori dei parametri di HDBSCAN sono i seguenti: *min-samples* = 1, *cluster-selection-epsilon* = 2.0505.
Sulla destra (vettori a 2 dimensioni) invece vengono identificati 5 cluster (di cui due molto piccoli), che sono simili a quelli originali. Il punteggio *V-Measure* è 0.920 e i valori dei parametri di HDBSCAN sono i seguenti: *min-samples* = 2, *cluster-selection-epsilon* = 1.8005. Come anticipato l'utilizzo di T-SNE ha messo in evidenza le dimensioni più significative dello spazio, facendone emergere i *cluster*.



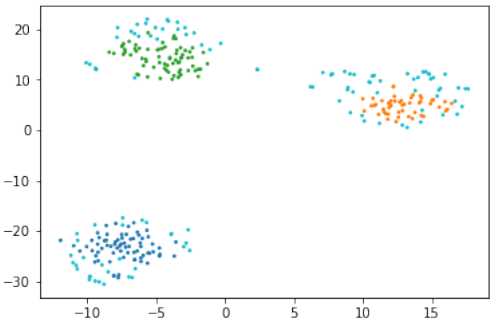
(a) *Cluster* identificati tra i vettori a 16 dimensioni.



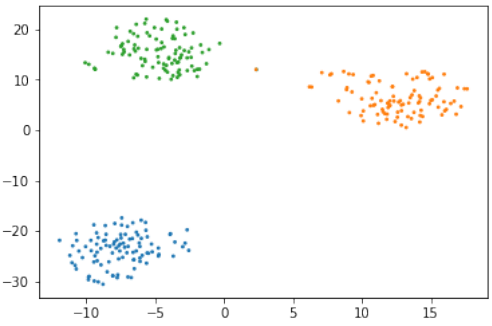
(b) *Cluster* identificati tra i vettori a 2 dimensioni calcolati con T-SNE.

Figura 4.9: Risultati di HDBSCAN sui vettori ottenuti da DDGK (senza normalizzazione).

Si è sperimentata anche l'applicazione di HDBSCAN sui vettori normalizzati. I vettori ottenuti da DDGK (in 16 dimensioni) sono stati normalizzati e da questi è stata calcolata la proiezione in 2 dimensioni usando T-SNE.



(a) *Cluster* identificati tra i vettori a 16 dimensioni.



(b) *Cluster* identificati tra i vettori a 2 dimensioni calcolati con T-SNE.

Figura 4.10: Risultati di HDBSCAN sui vettori ottenuti da DDGK **normalizzati**.

In figura Figura 4.10 sono illustrati i risultati.
Sulla sinistra (vettori a 16 dimensioni) si ottengono 4 *cluster*, in cui vengono correttamente identificati solamente i centri dei *cluster* di interesse. Il punteggio *V-Measure* è 0.586, poco più alto di quello precedente, e i valori dei parametri di HDBSCAN sono i seguenti: *min-samples* = 1, *cluster-selection-epsilon* = 0.0505.
Al contrario sulla destra (vettori a 2 dimensioni) vengono identificati che rispecchiano quasi perfettamente quelli originali. Il punteggio *V-Measure* è

0.991 e i valori dei parametri di HDBSCAN sono i seguenti: *min-samples* = 17, *cluster-selection-epsilon* = 2.9505.

```
eps_values = np.arange(0.0005, 3, 0.05).tolist()
min_samples_values = np.arange(1, 60).tolist()
best_score, best_args = 0, None
args_grid = itertools.product(eps_values, min_samples_values)
for eps, min_samples in args_grid:
    clusters = HDBSCAN(min_samples=min_samples,
        cluster_selection_epsilon=eps)
        .fit(embeddings).labels_
    score = v_measure_score(graph_labels, clusters)
    if score >= best_score:
        best_score, best_args = score, args
```

Listato 4.5: Codice per determinare i migliori parametri per HDBSCAN.

Valutazione qualitativa

Si presentano alcune delle coppie di eventi più vicine nello spazio vettoriale e si valuta manualmente la loro similarità semantica.

Figura 4.11: Un esempio di eventi molto simili dal dataset EPI.

In Figura 4.11 sono illustrati due eventi dal dataset *Epigenetics and Post-translational Modifications*, molto vicini nello spazio vettoriale. Si può osservare che per entrambi gli eventi possiedono lo stesso *trigger* e due argomenti con i medesimi ruoli. In particolare *H3* è una specifica famiglia di istoni (*histone*), mentre *T45* (anche *Threonine 45*) è una proteina della famiglia *serina/treonina protein-chinasi SGK* (*Ser / Thr*). Tutte le entità sono a due a due semanticamente simili, e così anche gli eventi stessi.

In Figura 4.12 sono rappresentati due semplici eventi dal dataset *Infectious Diseases*, anch'essi associati a vettori molto vicini tra loro. Il *trigger* in questo

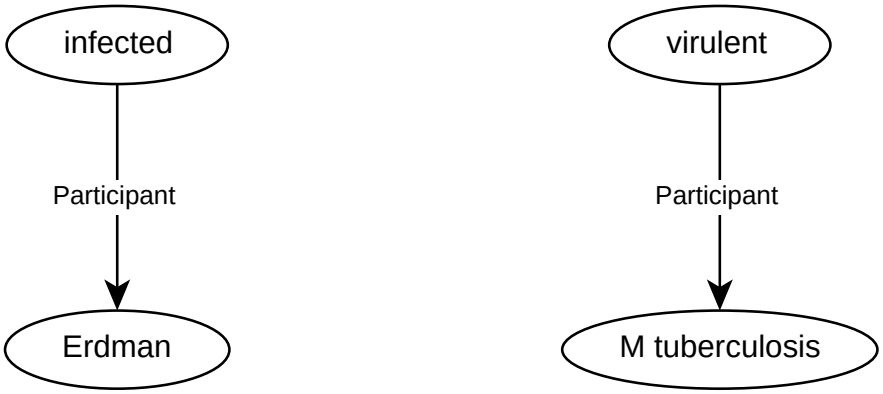


Figura 4.12: Un esempio di eventi molto simili dal dataset ID.

caso è differente, ma il significato è estremamente simile: entrambi i termini si riferiscono alla presenza di microorganismi nocivi. L'unico argomento ha lo stesso ruolo in entrambi gli eventi; in particolare *Erdman* è il nome di uno specifico ceppo del *Mycobacterium tuberculosis* (*M tuberculosis*). Anche in questo caso i due eventi esprimono un significato estremamente simile.

4.4.4 Osservazioni

L'obiettivo è quello di modellare in forma di eventi la conoscenza contenuta nella letteratura biomedica, in modo da poterli poi aggregare in un unico *knowledge graph* con tutti i vantaggi che ne conseguono (vedi Sezione 1.3). Una volta ottenuto un grafo è possibile fare interrogazioni e quantificare l'importanza di un termine o di una affermazione. La struttura proposta combina due soluzioni esistenti che risolvono parzialmente il problema in esame: DeepEventMine è in grado di estrarre dal testo singole unità di conoscenza in forma di eventi, fornendo anche una rappresentazione semantica delle entità e dei *trigger* come *embedding* ottenuti manipolando il risultato del *language model* utilizzato (in questo caso SciBERT); DDGK è ideato per la rappresentazione di grafi generici non necessariamente legati al NLP, inoltre considera che le etichette associate ai nodi siano "discrete" (facenti parte di un insieme predefinito) mentre in questo caso ad ogni nodo è associato un *embedding* che ne codifica il significato semantico. Entrambi risolvono solo una parte del problema di interesse, rispettivamente l'estrazione degli eventi e la loro aggregazione.

In questo specifico caso DeepEventMine viene utilizzato solamente per ottenere gli *embedding* per le entità, dato che viene applicato sugli stessi dataset su cui è stato addestrato. Al contrario con DDGK è stato necessario modificare

la funzione di *loss* per poter essere applicata a grafi i cui nodi possiedono attributi continui (in questo caso in uno spazio vettoriale): i risultati dimostrano l'efficacia di questa variazione nel preservare la semantica stessa associata ai nodi. Il modello così ottenuto può essere usato anche separatamente da DeepEventMine, per elaborare grafi con le caratteristiche descritte. Nonostante il modello non sia supervisionato in quanto non richiede dati etichettati, è comunque necessario determinare il valore ottimale degli iperparametri, in particolare il coefficiente C_a della funzione di *loss* per la coerenza degli attributi.

Capitolo 5

Conclusioni e sviluppi futuri

In questa tesi è stato esplorato il problema di estrazione di conoscenza strutturata da testo in linguaggio naturale. È stata proposta una soluzione concettuale, che poi è stata implementata valutando iterativamente l'applicazione di diverse soluzioni esistenti, adattate specificatamente per l'analisi di testo del dominio medico, formale o meno (come post pubblicati da pazienti e caregiver in un contesto di *social network*). Sono state inizialmente considerate metodologie ad ampio dominio ma con risultati limitati, via via sempre più specifiche ma più efficaci. Si è infine giunti all'utilizzo di **eventi** per catturare concetti semantici, che permettono una elevata espressività di significati anche molto complessi, mantenendo comunque una struttura solida e ben definita. La loro rappresentazione come **grafi** apre ampie possibilità per l'utilizzo di una vasta gamma di strumenti dedicati alla loro analisi, comparazione e aggregazione. In questo caso viene usato un algoritmo non supervisionato per calcolarne gli *embedding* in un nuovo spazio vettoriale in cui viene preservato opportunamente il loro significato semantico. I risultati ottenuti, seppur solo preliminari, dimostrano le potenzialità dei metodi utilizzati.

Il metodo di valutazione adoperato è significativo ma non esaustivo. La separabilità dei vettori con iperpiani e la formazione di *cluster* che rispecchiano l'appartenenza degli eventi a diversi dataset, dimostra la capacità della struttura a grafo di catturare la semantica dell'evento e che la soluzione proposta sia in grado di sfruttarla e codificarla efficacemente. Allo stesso tempo non si è in grado di valutare la qualità dei risultati a un livello più sottile: la mancanza di dati etichettati non permette di valutare automaticamente la correttezza delle "distanze" tra gli eventi nello spazio vettoriale; nonostante sia possibile effettuarla in modo semi-supervisionato, per ragioni di tempo non è stato possibile metterla in pratica. Una verifica manuale sarebbe possibile ma costosa: dal momento che il testo analizzato fa riferimento a un dominio biomedico altamente specifico, una valutazione manuale esaustiva richiede il coinvolgimenti

di esperti con competenze adeguate e risorse che vanno oltre gli obiettivi di questa tesi.

Il primo fronte di sviluppo quindi è senz'altro un **perfezionamento e arricchimento della metodologia di valutazione**, che può essere ottenuto in diversi modi.

- Per valutare individualmente la rappresentazione dei grafi nello spazio vettoriale è possibile codificare eventi *"gold standard"*, cioè etichettati da ricercatori umani e quindi la cui correttezza è certa. È comunque necessario utilizzare DeepEventMine per ottenere gli *embedding* semantici delle entità, ma selezionando solamente gli eventi predetti correttamente.

- Utilizzare tutti i sette dataset disponibili invece che soltanto tre permette di valutare più accuratamente le capacità del modello.

- Per verificare la correttezza delle similarità tra singoli eventi è possibile usare un approccio semi-supervisionato: si sfruttano modelli per l'equivalenza semantica tra frasi (vedi Sezione 1.5) per ottenere un valore di similarità tra le frasi da cui gli eventi sono stati estratti, e si definisce una soglia sopra la quale le due frasi vengono considerate equivalenti. Si cerca poi una seconda soglia da applicare alla distanza tra coppie di eventi nello spazio vettoriale, in modo da ottenere un risultato il più simile possibile a quello tra le frasi. In questo modo è possibile valutare quanto la distanza tra eventi rispecchia la similarità delle frasi da cui provengono. Anche se questo metodo possiede alcune limitazioni (ci si basa sulla similarità calcolata da un altro modello che potrebbe non essere accurata e non è detto che due frasi siano sempre simili anche se gli eventi estratti potrebbero esserlo) permette di avere una valutazione oggettiva del risultato senza bisogno di dati etichettati.

Inoltre per ottenere migliori risultati è possibile effettuare diversi esperimenti:

- considerare diversi valori dell'iperparametro C_a per determinare quale di essi ottiene risultati migliori, in modo da assegnare la giusta importanza alla componente strutturale e a quella semantica;

- utilizzare un metodo più sofisticato per la scelta dei prototipi, che al momento vengono selezionati in modo casuale, ad esempio basato su euristiche per scegliere i grafi più diversi fra loro.

Dopo aver verificato la capacità del modello di catturare la similarità tra eventi, è possibile usarla per la loro aggregazione.

In primo luogo individuando gli eventi che esprimono la stessa interazione o interazioni estremamente simili è possibile quantificare l'importanza e la veridicità di una certa affermazione: soprattutto in ambito medico è fondamentale accompagnare i risultati ottenuti da reti profonde (in questo caso gli eventi estratti) con valutazioni statistiche che permettano di verificarne la correttezza (*explainability*). Usando lo spazio vettoriale in cui sono rappresentati gli eventi è possibile identificare piccoli *cluster* fortemente coesi in cui viene espresso lo stesso concetto semantico, e contare la dimensione di questi *cluster* quantificando quindi il suo numero di occorrenze nel testo.

In secondo luogo, sfruttando l'*attention* inter-grafo utilizzata da DDGK è possibile aggregare gli eventi con granularità molto fine (a livello di singoli nodi), fondendoli per ottenere strutture più complesse e più espressive. A loro volta queste potranno essere combinate, identificando i nodi comuni anche tra eventi con semantica differente, in un unico *knowledge graph* permettendo di effettuare interrogazioni più naturali, calcolare statistiche utili e applicare tali contributi anche per l'apprendimento dinamico di *knowledge graph* nel tempo. Usando strumenti e algoritmi noti nella teoria dei grafi è possibile determinare l'importanza delle entità nel *knowledge graph* (ad esempio usando algoritmi per il calcolo della centralità, vedi Figura 5.1), oppure le relazioni indirette tra di esse (ad esempio usando algoritmi di esplorazione del grafo).

In conclusione, si è mosso un primo importante passo per il raggiungimento del complesso obiettivo discusso in questa tesi. Un utilizzo diretto sui post richiederà in futuro un riaddestramento di DeepEventMine con un dataset più in linea con i dati utilizzati, in modo da catturare infine la voce dei pazienti.

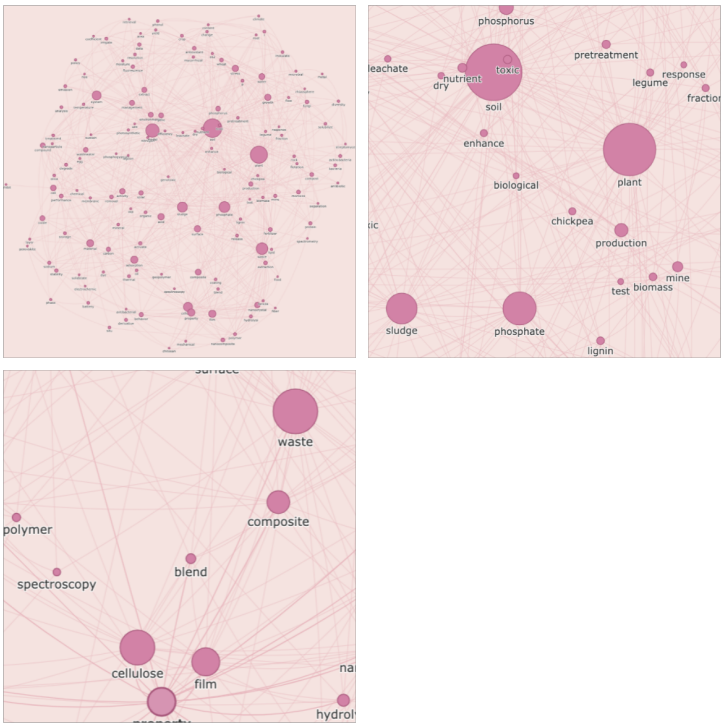


Figura 5.1: Con statistiche strutturali è possibile determinare l'importanza dei singoli nodi all'interno del *knowledge graph*, in questo caso rispecchiata dalla loro dimensione.

Fonte: `https://towardsdatascience.com/temporal-semantic-network-analysis-bd8869c10f10`

Ringraziamenti

Ringrazio i miei relatori, il professor Moro, la professoressa Carbonaro e il dottor Frisoni, per avermi guidato nella realizzazione di questa tesi con la loro competenza e professionalità. A Giacomo faccio un ringraziamento particolare per la passione con cui si dedica al proprio lavoro e ad aiutare gli altri.

Ringrazio la mia ragazza Sara che mi ha spronato durante la stesura, sostenendomi, consigliandomi e nutrendomi.

Infine un grande grazie alla mia famiglia, che mi ha sempre guidato e accompagnato con affetto nella grande avventura della vita che mi ha condotto fino a qui.

Bibliografia

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[2] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112, 2014.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[4] A. Radford and Karthik Narasimhan. Improving language understanding by generative pre-training, 2018.

[5] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[6] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how BERT works. *Trans. Assoc. Comput. Linguistics*, 8:842–866, 2020.

[7] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3613–3618. Association for Computational Linguistics, 2019.

[8] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinform.*, 36(4):1234–1240, 2020.

[9] Jari Björne et al. *Biomedical event extraction with machine learning.* TUCS Dissertations, 2014.

[10] Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176, Beijing, China, July 2015. Association for Computational Linguistics.

[11] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL, 2014.

[12] Trung Minh Nguyen and Thien Huu Nguyen. One for all: Neural joint modeling of entities and events. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6851–6858. AAAI Press, 2019.

[13] Hai-Long Trieu, Thy Thy Tran, Khoa N A Duong, Anh Nguyen, Makoto Miwa, and Sophia Ananiadou. DeepEventMine: end-to-end neural nested event extraction from biomedical texts. *Bioinformatics*, 36(19):4910–4917, 06 2020.

[14] Mark Craven and Johan Kumlien. Constructing biological knowledge bases by extracting information from text sources. In Thomas Lengauer, Reinhard Schneider, Peer Bork, Douglas L. Brutlag, Janice I. Glasgow, Hans-Werner Mewes, and Ralf Zimmer, editors, *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology, August 6-10, 1999, Heidelberg, Germany*, pages 77–86. AAAI, 1999.

[15] Chenguang Wang, Xiao Liu, and Dawn Song. Language models are open knowledge graphs, 2021.

[16] Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. Matching the blanks: Distributional similarity for relation learning. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2895–2905. Association for Computational Linguistics, 2019.

[17] Wenxuan Zhou, Kevin Huang, Tengyu Ma, and Jing Huang. Document-level relation extraction with adaptive thresholding and localized context pooling. *CoRR*, abs/2010.11304, 2020.

[18] Xiaotian Jiang, Quan Wang, Peng Li, and Bin Wang. Relation extraction with multi-instance multi-label convolutional neural networks. In Nicoletta Calzolari, Yuji Matsumoto, and Rashmi Prasad, editors, *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 1471–1480. ACL, 2016.

[19] Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. Classifying relations via long short term memory networks along shortest dependency paths. In Lluís Màrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1785–1794. The Association for Computational Linguistics, 2015.

[20] Ryuichi Takanobu, Tianyang Zhang, Jiexi Liu, and Minlie Huang. A hierarchical framework for relation extraction with reinforcement learning.

In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 7072–7079. AAAI Press, 2019.

[21] Yucheng Wang, Bowen Yu, Yueyang Zhang, Tingwen Liu, Hongsong Zhu, and Limin Sun. Tplinker: Single-stage joint extraction of entities and relations through token pair linking. In Donia Scott, Núria Bel, and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 1572–1582. International Committee on Computational Linguistics, 2020.

[22] Zexuan Zhong and Danqi Chen. A frustratingly easy approach for joint entity and relation extraction. *CoRR*, abs/2010.12812, 2020.

[23] Haoyu Wang, Ming Tan, Mo Yu, Shiyu Chang, Dakuo Wang, Kun Xu, Xiaoxiao Guo, and Saloni Potdar. Extracting multiple-relations in one-pass with pre-trained transformers. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1371–1377. Association for Computational Linguistics, 2019.

[24] Amit Singhal. Introducing the knowledge graph: things, not strings, May 2012.

[25] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. In Michael Martin, Martí Cuquet, and Erwin Folmer, editors, *Joint Proceedings of the Posters and Demos Track of the 12th International Conference on Semantic Systems - SEMANTiCS2016 and the 1st International Workshop on Semantic Change & Evolving Semantics (SuCCESS'16) co-located with the 12th International Conference on Semantic Systems (SEMANTiCS 2016), Leipzig, Germany, September 12-15, 2016*, volume 1695 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016.

[26] Zhihao Fan, Yeyun Gong, Zhongyu Wei, Siyuan Wang, Yameng Huang, Jian Jiao, Xuanjing Huang, Nan Duan, and Ruofei Zhang. An enhanced knowledge injection model for commonsense generation. In Donia Scott, Núria Bel, and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona,*

Spain (Online), December 8-13, 2020, pages 2014–2025. International Committee on Computational Linguistics, 2020.

[27] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. K-BERT: enabling language representation with knowledge graph. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 2901–2908. AAAI Press, 2020.

[28] Tianxiang Sun, Yunfan Shao, Xipeng Qiu, Qipeng Guo, Yaru Hu, Xuanjing Huang, and Zheng Zhang. Colake: Contextualized language and knowledge embedding. In Donia Scott, Núria Bel, and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 3660–3670. International Committee on Computational Linguistics, 2020.

[29] Bin He, Di Zhou, Jinghui Xiao, Xin Jiang, Qun Liu, Nicholas Jing Yuan, and Tong Xu. Integrating graph contextualized knowledge into pre-trained language models. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pages 2281–2290. Association for Computational Linguistics, 2020.

[30] Yu Chen, Lingfei Wu, and Mohammed J. Zaki. Bidirectional attentive memory networks for question answering over knowledge bases. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2913–2923. Association for Computational Linguistics, 2019.

[31] Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. Cognitive graph for multi-hop reading comprehension at scale. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2694–2703. Association for Computational Linguistics, 2019.

[32] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. Multi-task feature learning for knowledge graph enhanced

recommendation. In Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia, editors, *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 2000–2010. ACM, 2019.

[33] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. KGAT: knowledge graph attention network for recommendation. In Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis, editors, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 950–958. ACM, 2019.

[34] Rita T. Sousa, Sara Silva, and Catia Pesquita. Evolving knowledge graph similarity for supervised learning in complex biomedical domains. *BMC Bioinform.*, 21(1):6, 2020.

[35] Boran Hao, Henghui Zhu, and Ioannis Ch. Paschalidis. Enhancing clinical BERT embedding using a biomedical knowledge base. In Donia Scott, Núria Bel, and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 657–661. International Committee on Computational Linguistics, 2020.

[36] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8(3):489–508, 2017.

[37] Sabbir Rashid, Eva Blomqvist, Cogan Matthew Shimizu, and Michel Dumontier. On the creation of knowledge graphs: A report on best practices and their future. *Dagstuhl Reports*, 8(9):49–53, 2018.

[38] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. A survey on knowledge graphs: Representation, acquisition and applications. *CoRR*, abs/2002.00388, 2020.

[39] Marvin Hofer, Sebastian Hellmann, Milan Dojchinovski, and Johannes Frey. The new dbpedia release cycle: Increasing agility and efficiency in knowledge extraction workflows. In Eva Blomqvist, Paul Groth, Victor de Boer, Tassilo Pellegrini, Mehwish Alam, Tobias Käfer, Peter Kieseberg, Sabrina Kirrane, Albert Meroño-Peñuela, and Harshvardhan J. Pandit, editors, *Semantic Systems. In the Era of Knowledge Graphs - 16th International Conference on Semantic Systems, SEMANTiCS 2020, Amsterdam, The Netherlands, September 7-10, 2020, Proceedings*, volume 12378 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2020.

[40] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2787–2795, 2013.

[41] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 1112–1119. AAAI Press, 2014.

[42] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam, editors, *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607. Springer, 2018.

[43] Baoxu Shi and Tim Weninger. Open-world knowledge graph completion. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 1957–1964. AAAI Press, 2018.

[44] Ni Lao and William W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Mach. Learn.*, 81(1):53–67, 2010.

[45] Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. Chains of reasoning over entities, relations, and text using recurrent neural networks. In Mirella Lapata, Phil Blunsom, and Alexander Koller, editors, *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 132–141. Association for Computational Linguistics, 2017.

[46] Wenhan Xiong, Thien Hoang, and William Yang Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. In Martha

Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 564–573. Association for Computational Linguistics, 2017.

[47] Yelong Shen, Jianshu Chen, Po-Sen Huang, Yuqing Guo, and Jianfeng Gao. M-walk: Learning to walk over graphs using monte carlo tree search. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6787–6798, 2018.

[48] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In Kevin Knight, Ani Nenkova, and Owen Rambow, editors, *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 260–270. The Association for Computational Linguistics, 2016.

[49] Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. ERNIE 2.0: A continual pre-training framework for language understanding. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8968–8975. AAAI Press, 2020.

[50] Yukun Ma, Erik Cambria, and Sa Gao. Label embedding for zero-shot fine-grained named entity typing. In Nicoletta Calzolari, Yuji Matsumoto, and Rashmi Prasad, editors, *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 171–180. ACL, 2016.

[51] Junheng Hao, Muhao Chen, Wenchao Yu, Yizhou Sun, and Wei Wang. Universal representation learning of knowledge bases by jointly embedding instances and ontological concepts. In Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis, editors, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 1709–1719. ACM, 2019.

[52] Octavian-Eugen Ganea and Thomas Hofmann. Deep joint entity disambiguation with local neural attention. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2619–2629. Association for Computational Linguistics, 2017.

[53] Charu C. Aggarwal. Graph clustering. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning and Data Mining*, pages 570–579. Springer, 2017.

[54] Charu C. Aggarwal and Haixun Wang. A survey of clustering algorithms for graph data. In Charu C. Aggarwal and Haixun Wang, editors, *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, pages 275–301. Springer, 2010.

[55] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. Graph clustering based on structural/attribute similarities. *Proc. VLDB Endow.*, 2(1):718–729, 2009.

[56] Guixiang Ma, Lifang He, Bokai Cao, Jiawei Zhang, Philip S. Yu, and Ann B. Ragin. Multi-graph clustering based on interior-node topology with applications to brain networks. In Paolo Frasconi, Niels Landwehr, Giuseppe Manco, and Jilles Vreeken, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I*, volume 9851 of *Lecture Notes in Computer Science*, pages 476–492. Springer, 2016.

[57] HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Trans. Knowl. Data Eng.*, 30(9):1616–1637, 2018.

[58] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowl. Based Syst.*, 151:78–94, 2018.

[59] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In James Bailey, Alistair Moffat, Charu C. Aggarwal, Maarten de Rijke, Ravi Kumar, Vanessa Murdock, Timos K. Sellis, and Jeffrey Xu Yu, editors, *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, pages 891–900. ACM, 2015.

[60] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani, editors, *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 701–710. ACM, 2014.

[61] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[62] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991.

[63] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80, 2009.

[64] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Networks Learn. Syst.*, 32(1):4–24, 2021.

[65] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[66] Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. Simgnn: A neural network approach to fast graph similarity computation. In J. Shane Culpepper, Alistair Moffat, Paul N. Bennett, and Kristina Lerman, editors, *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*, pages 384–392. ACM, 2019.

[67] Robert P. W. Duin and Elzbieta Pekalska. The dissimilarity space: Bridging structural and statistical pattern recognition. *Pattern Recognit. Lett.*, 33(7):826–832, 2012.

[68] Xinbo Gao, Bing Xiao, Dacheng Tao, and Xuelong Li. A survey of graph edit distance. *Pattern Anal. Appl.*, 13(1):113–129, 2010.

[69] John W. Raymond and Peter Willett. Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *J. Comput. Aided Mol. Des.*, 16(7):521–533, 2002.

[70] Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36(3):1171 – 1220, 2008.

[71] Rami Al-Rfou, Bryan Perozzi, and Dustin Zelle. DDGK: learning graph representations for deep divergence graph kernels. In Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia, editors, *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 37–48. ACM, 2019.

[72] Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119, 2013.

[73] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[74] Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. Sentence similarity learning by lexical decomposition and composition. In Nicoletta Calzolari, Yuji Matsumoto, and Rashmi Prasad, editors, *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 1340–1349. ACL, 2016.

[75] Huy Nguyen Tien, Minh Nguyen Le, Yamasaki Tomohiro, and Izuha Tatsuya. Sentence modeling via multiple word embeddings and multi-level comparison for semantic textual similarity. *Inf. Process. Manag.*, 56(6), 2019.

[76] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International*

Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019, pages 3980–3990. Association for Computational Linguistics, 2019.

[77] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 670–680. Association for Computational Linguistics, 2017.

[78] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[79] Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In Steven Bethard, Marine Carpuat, Marianna Apidianaki, Saif M. Mohammad, Daniel M. Cer, and David Jurgens, editors, *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017*, pages 1–14. Association for Computational Linguistics, 2017.

[80] Giacomo Frisoni, Gianluca Moro, and Antonella Carbonaro. Learning interpretable and statistically significant knowledge from unlabeled corpora of social text messages: A novel methodology of descriptive text mining. In Slimane Hammoudi, Christoph Quix, and Jorge Bernardino, editors, *Proceedings of the 9th International Conference on Data Science, Technology and Applications, DATA 2020, Lieusaint, Paris, France, July 7-9, 2020*, pages 121–132. SciTePress, 2020.

[81] Giacomo Frisoni. A new unsupervised methodology of Descriptive Text Mining for Knowledge Graph Learning. Master thesis, University of Bologna, 2020.

[82] Giacomo Frisoni and Gianluca Moro. Phenomena explanation from text. In *Data Management Technologies and Applications - 9th International Conference, DATA, Revised Selected Papers*, Communications in Computer and Information Science, pages 1–24. Springer, 2021. Accepted as regular paper.

[83] Giacomo Frisoni, Gianluca Moro, and Antonella Carbonaro. Unsupervised descriptive text mining for knowledge graph learning. In Ana L. N. Fred and Joaquim Filipe, editors, *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, IC3K 2020, Volume 1: KDIR, Budapest, Hungary, November 2-4, 2020*, pages 316–324. SCITEPRESS, 2020.

[84] Giacomo Frisoni, Gianluca Moro, and Antonella Carbonaro. Towards rare disease knowledge graph learning from social posts of patients. In Anna Visvizi, Miltiadis D. Lytras, and Naif R. Aljohani, editors, *Research and Innovation Forum 2020 - Disruptive Technologies in Times of Change, RIIFORUM 2020, Athens, Greece, 15-17 April 2020*, pages 577–589. Springer, 2020.

[85] Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. Docred: A large-scale document-level relation extraction dataset. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 764–777. Association for Computational Linguistics, 2019.

[86] Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4803–4809. Association for Computational Linguistics, 2018.

[87] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[88] Ricardo J. G. B. Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu, editors, *Advances in Knowledge Discovery and Data Mining, 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, April 14-17, 2013, Proceedings, Part II*, volume 7819 of *Lecture Notes in Computer Science*, pages 160–172. Springer, 2013.

[89] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In Jason Eisner, editor, *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on*

Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic, pages 410–420. ACL, 2007.