POLITECNICO DI TORINO

Corso di Laurea Specialistica in
Ingegneria Informatica

Tesi di Laurea Specialistica

Indicizzazione di videocorsi basata su tecniche di Named Entity Disambiguation

Relatori

Farinetti Laura
Cagliero Luca

Candidato

Degiovanni Alex

Dicembre, 2018

Riconoscimenti

È con sollievo e felicità che mi accingo a scrivere questi ringraziamenti, frutto di 5 anni di sacrifici, in cui però ho imparato veramente tanto. Vorrei spendere due parole di ringraziamento nei confronti di tutte le persone che mi hanno sostenuto e aiutato durante questo percorso.
Un ringraziamento particolare è doveroso farlo ai miei relatori, i professori Laura Farinetti e Luca Cagliero, insieme al dottorando Lorenzo Canale con cui ho lavorato in questi mesi, che mi hanno dato consigli preziosi.
Mi avete fornito tutti gli strumenti di cui avevo bisogno per intraprendere la strada giusta e portare a compimento la mia tesi.
Un grande ringraziamento a mia madre e mio padre che, con il loro dolce e instancabile sostegno, sia morale che economico, hanno contribuito alla mia formazione personale e mi hanno permesso di arrivare oggi davanti a questa commissione. Quando penso ai genitori della Mulino Bianco io penso a voi.
Alla mia fidanzata, che mi ha sostenuto ed è stata la mia ancora nei momenti più difficili. Grazie per essere stata sempre al mio fianco e per aver affrontato, insieme a me, ogni problema che si è presentato nel corso di questi anni.
Per ultimi, ma non meno importanti, i miei amici e compagni di università: sempre presenti nei momenti di sconforto, pronti a gioire nei momenti di gioia.
Grazie!

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Politecnico di Torino offre numerosi servizi per andare incontro alle esigenze degli studenti durante il loro percorso di studi. Uno di questi è sicuramente il servizio di videolezioni online, offerto da numerosi indirizzi. In questo contesto attualmente manca una funzionalità di ricerca veloce con cui uno studente, scrivendo determinate *keyword* in una barra di ricerca, possa ritrovare velocemente tutti i momenti in cui il professore ha fatto riferimento all'argomento cercato. La tesi si inquadra all'interno di un progetto di ricerca mirato a sviluppare un'applicazione in grado di fornire un servizio di indicizzazione per corsi videoregistrati all'interno del Politecnico di Torino. In generale l'obiettivo è quello di riconoscere gli argomenti principali all'interno di un video, basandosi sia sull'audio, che sui contenuti proiettati (slide, fogli ecc..). Effettuare una divisione rigida di segmenti video e assegnare ad ognuno di essi un *topic* principale è un approccio poco efficiente, in quanto durante una spiegazione il professore tende a fare riferimenti a più argomenti, perciò etichettare un segmento con un solo topic principale sarebbe limitante ai fini del progetto. Avendo a disposizione il *transcript* del corso (cioé la trascrizione delle parole associate all'istante temporale in cui sono state pronunciate), l'approccio scelto consiste nell'identificare delle entità (cioè unità di informazioni) all'interno di esso. Identificando questa lista di entità è possibile, a seguito di una ricerca da parte di uno studente e l'assegnazione di essa all'entità che più la rappresenta, trovare velocemente tutti i riferimenti temporali all'interno del corso in cui l'argomento ricercato è stato trattato. Dato il testo del parlato, per ritrovare le entità corrette al suo interno, è necessario applicare delle procedure di *Named entity disambiguation* (NED).
Per NED si intende il processo di identificazione di entità all'interno di un testo non strutturato, collegando le entità ad una base di conoscenza (Wikipedia, DBpedia, Wikidata ecc..). Questa operazione è resa più difficile dalla forte ambiguità del linguaggio, in quanto una qualsiasi parola può avere significati diversi a seconda del contesto in cui viene utilizzata, e di conseguenza, può riferirsi a entità completamente opposte. L'obiettivo del modello che si andrà a realizzare è quello di riuscire ad identificare il maggior numero di identità corrette all'interno dei transcript delle videolezioni, relative effettivamente al corso in questione. Alcune problematiche rendono questa ricerca più complicata: i transcript utilizzati contengono molti errori sintattici (trascrizione non corretta di parole) ed inoltre, non è presente alcun segno di punteggiatura che rende impossibile identificare frasi diverse, e di conseguenza concetti logici diversi.
In letteratura esistono dei sistemi pubblici che effettuano *disambiguation*. Questi

servizi (estrattori) sono stati progettati per funzionare bene in contesti generici, ed utilizzando come input dei testi puliti con ampio utilizzo della punteggiatura. Utilizzando GERBIL, un tool online che permette di confrontare gli estrattori esistenti tra loro, si è verificato il loro funzionamento in presenza dei transcript presi in considerazione. Il risultato è stato pessimo, dovuto oltre che all'input non perfetto, anche alla lingua italiana, infatti, in genere, questi strumenti tendono a funzionare meglio in inglese rispetto ad altre lingue. Nel modello che si è andato a realizzare sono stati seguiti dei passi differenti rispetto alla letteratura. Gli estrattori online si basano principalmente su Wikipedia e DBpedia come base di conoscenza, mentre il modello da noi sviluppato utilizza come fonte Wikidata. Questa base di conoscenza presenta principalmente due pregi: essa è strutturata e multilingua. Strutturata perché i suoi contenuti sono presentati nel formato standard RDF in triple, composte da *soggetto*, *predicato* e *oggetto*, dove il soggetto corrisponde ad una entità, il predicato è paragonabile ad una proprietà specifica, e l'oggetto corrisponde al valore corrispondente. Multilingua poiché Wikidata non ha versioni differenti, una per ogni lingua (come Wikipedia), ma in quanto associate alla stessa entità sono presenti tutte le descrizioni disponibili riferite ad ogni lingua presente. Queste descrizioni corrispondono ad etichette: alcuni esempi sono l'etichetta principale (il nome che la rappresenta), le etichette alternative (sinonimi del nome), le descrizioni dell'entità.
Come punto iniziale nello sviluppo del modello proposto, scegliendo come riferimento la lingua italiana e quella inglese, sono state salvate tutte le entità presenti in Wikidata in locale, utilizzando *mongoDB* come database non relazionale. Per ogni entità sono state salvate le etichette principali nelle due lingue, arrivando ad avere più di 40 milioni di stringhe. Per quanto riguarda i transcript di partenza, dopo che si è diviso l'input in singoli token (parole singole), senza tener conto degli spazi tra le parole e delle *stopword*, sono state generate delle stringhe di riferimento di lunghezza variabile, tra 1 e 10 token ordinati. In seguito, per ogni stringa generata, si è effettuata una ricerca all'interno delle etichette di tutte le entità di Wikidata salvate in precedenza, tenendo in considerazione solo le label sufficientemente *simili* alla stringa di riferimento (cioè simili sintatticamente). Dopo questa fase, ad ogni stringa sarà associato un numero variabile di label, da 0 a n, dalle quali è stato possibile risalire alle entità corrispondenti, dette entità *candidate*. Infine, si è effettuata la scelta dell'entità candidata più opportuna, utilizzando due alberi di decisione. Per discriminare tra i candidati, questi *Decision Tree*, oltre alla similarità tra label e stringa di partenza, utilizzano due valori aggiuntivi molto importanti. Questi due valori corrispondono alla similarità dell'entità candidata rispetto all'entita principale (quella che rappresenta il corso preso in considerazione) nel grafo di Wikidata e Wikipedia. Quest'ultime esprimono una misura di *vicinanza* tra le entità e il contesto preso in considerazione. Ricapitolando, è stato utilizzato un approccio locale per la scelta dei candidati basato sulla vicinanza tra stringhe utilizzando Wikidata. Per la fase di *disambiguation* è stato utilizzato un approccio globale, tenendo in considerazione anche il contesto attorno alla menzione per la scelta dell'entità più opportuna.
Per la valutazione del funzionamento del modello, oltre a venire testato su alcuni frammenti provenienti dalle videolezioni annotati utilizzando Brat (un tool che fornisce un interfaccia grafica per eseguire annotazioni su testi), il test è stato eseguito anche su un set di dati composto da alcuni tweet di Twitter, utilizzati in EVALITA 2016. EVALITA è una campagna periodica di valutazione di *Natural Language*

Processing (NLP) e strumenti vocali per la lingua italiana.
Analizzando i risultati ottenuti si è potuto apprendere quanto segue: il modello costruito è stato in grado di lavorare bene nell'ambito delle videolezioni, essendo in grado di disambiguare efficacemente in base soprattutto ai valori di similarità dei grafi di Wikidata e Wikipedia che sono stati la chiave per la realizzazione del modello. Gli estrattori presenti in letteratura hanno riscontrato numerose difficoltà nel lavorare in questo contesto. All'interno di questi frammenti sono presenti, oltre che entità inerenti al corso, anche entità generali, perciò per effettuare una verifica ulteriore queste ultime sono state filtrate ottenendo una *ground truth* contenente solo entità utili ai fini del corso. Anche in questo caso caso il modello si è dimostrato efficace, riuscendo a disambiguare la maggior parte delle entità.
Infine si è testato il modello sul dataset di Twitter. Come da previsioni, in questo contesto generale, in assenza dei valori di similarità provenienti da Wikidata e Wikipedia, il modello non è in grado di disambiguare correttamente, ottenendo però buoni risultati in quanto a *recall*.
Queste valutazioni sono state necessarie per mostrare pregi e difetti del lavoro realizzato, e per valutare possibili miglioramenti applicabili in futuro.

1.1 Struttura del report

Il lavoro di tesi è organizzato nel seguente modo:

- **Capitolo 2**. Un breve accenno sul contesto di lavoro e sugli obiettivi finali da raggiungere;
- **Capitolo 3**. Analisi della letteratura ed in particolare a quella sulla *Named entity disambiguation*;
- **Capitolo 4**. Analisi degli strumenti presenti online per fare disambiguazione e analisi di GERBIL, strumento utilizzato per confrontare estrattori tra loro;
- **Capitolo 5**. Descrizione dei passi seguiti per realizzare un primo modello di estrattore che fosse in grado di *lavorare* bene all'interno dei transcript;
- **Capitolo 6**. Creazione della *ground truth* che verrà utilizzata per valutare il comportamento del modello creato;
- **Capitolo 7**. Valutazione dei risultati ottenuti;
- **Capitolo 8**. Conclusioni e possibili lavori futuri di miglioramento.

Capitolo 2

Contesto

Il Politecnico di Torino fornisce molti servizi per aiutare gli studenti nel loro percorso di studi durante gli anni di università. Uno dei più importanti è sicuramente il servizio di videolezioni online, offerto sempre più negli ultimi anni. Grazie a questo, gli studenti sono agevolati non solo nel seguire i corsi comodamente da casa in caso di malattia o in caso di impegni extra-scolastici, ma anche durante la fase di studio, nel caso in cui alcuni argomenti non fossero chiari. La fig. 2.1 rappresenta la tipica pagina principale, a cui gli studenti, tramite il Portale della Didattica sulla propria pagina, possono accedere comodamente.

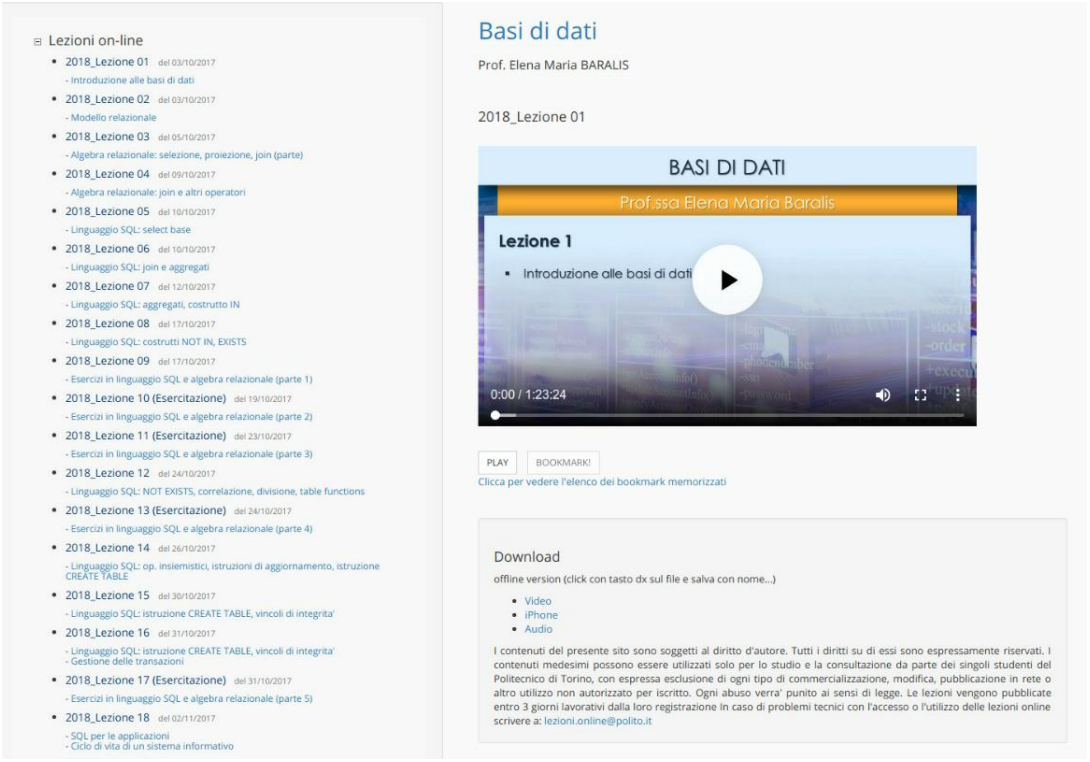


Figura 2.1: Immagine proveniente dal sito del Politecnico di torino, riguardante il corso *Basi di dati*.

La fig. 2.1 è stata presa dal sito del Politecnico di Torino[1], nel corso *Basi di dati*[2],

[1] `https://didattica.polito.it/`

[2] `https://didattica.polito.it/portal/pls/portal/sviluppo.guide.visualizza?p\_cod\_ins=14AFQPD&p\_a\_acc=2017`

corso offerto per Ingegneri Gestionali al secondo anno di triennale. Su questa pagina gli studenti possono liberamente scegliere la lezione che preferiscono e rivedere le parti di maggior interesse ai fini dello studio. Inoltre sono anche fornite altre funzionalità, tra le quali la possibilità di descrivere una determinata lezione, in modo da aiutare gli altri studenti che hanno accesso alla stessa pagina, oppure di scaricare comodamente le videolezioni per poterle vedere offline. In tutto questo manca una funzionalità che si potrebbe definire *essenziale*: una barra di ricerca che permetta di cercare contenuti specifici all'interno del corso. In questo modo infatti, uno studente potrebbe ricercare uno specifico argomento scrivendo alcune *keyword*, ottenenendo come risultato un elenco di link, corrispondenti ai momenti precisi in cui il professore ha parlato dell'argomento cercato. In generale infatti, nella fase finale di studio, a pochi giorni dall'esame, gli studenti sono interessanti a chiarirsi le idee su specifici argomenti, e solitamente non si ha il tempo di andare a cercare tutti i momenti in cui questi sono stati effettivamente trattati, rischiando così di non ripassare parti essenziali. Il servizio che si vuole realizzare va incontro proprio a questa esigenza. L'idea è quella di dividere ogni lezione in blocchi, caratterizzati da un argomento specifico, in modo che, nel momento in cui venga cercato un determinato argomento, lo studente possa vedere tutte le sezioni riguardanti quest'ultimo. Dividere una lezione in blocchi rigidi, dove ogni blocco corrisponde ad un topic specifico, non è appropriato, infatti generalmente durante una spiegazione, i professori tendono a fare riferimenti a molti argomenti, necessari per capire il tutto. Per questo motivo, l'obiettivo è stato quello di trovare un modo per riconoscere all'interno del video i concetti chiave, collegati il più possibile alla materia in questione. L'approccio scelto è stato quello di, partendo dalla trascrizione del parlato del professore (*transcript*) e avendo ogni riferimento tra parola e istante di tempo in cui è stata pronunciata, analizzare questo testo alla ricerca di concetti chiavi. In questo ambito, questi concetti principali verranno chiamati *entità*. In fig. 2.2 è riassunto il contesto. Il video è

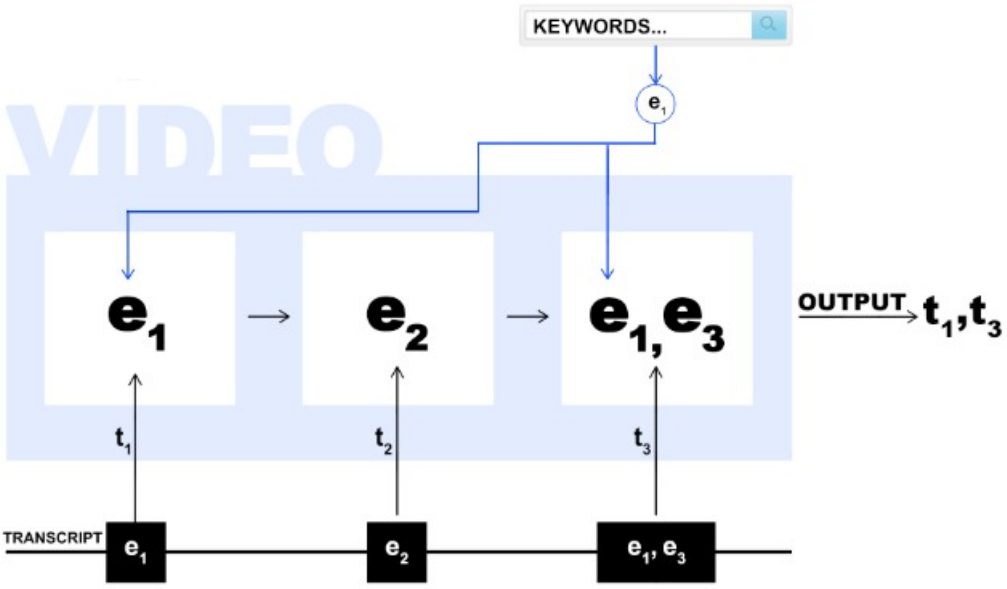


Figura 2.2: Esempio pratico di ricerca con relativo output.

composto da tanti spezzoni (chiamati *segmenti video*), ai quali sono associate deter-

minate parole pronunciate in quegli istanti, rappresentate dalla linea orizzonatale al fondo dell'immagine, cioè appunto dal transcript. Analizzando questo testo perciò, è possibile identificare determinate entità e, associate ai corrispondenti istanti di tempo t. Come detto prima, è possibile che nello stesso range temporale, vengano citate diverse entità, come nel caso di t_3 con e_1 ed e_3. A questo punto, nel momento in cui uno studente ricercasse un argomento, utilizzando l'entità che meglio lo rappresenta, allora potrebbe vedere vedere tutti gli istanti temporali corrispondenti a quest'ultima, e facilmente cercare all'interno di essi la parte di interesse. Nella figura 2.2, a seguito di una ricerca da parte di uno studente, viene identificata come appartenente all'entità e_1, perciò la risposta del sistema sarebbero gli istanti di tempo t_1 e t_3. Inoltre, è possibile che ad una ricerca corrispondano più entità, di conseguenza il risultato sarebbero tutti gli istanti di tempo delle singole e identificate.
Questo schema fa ampio utilizzo dei transcript, i quali sono stati generati utilizzando un servizio offerto da YouTube. Analizzando questi testi però, è facile notare la completa assenza di punteggiatura, e la trascrizione non corretta di alcune parole. Si prenda in considerazione il seguente frammento ottenuto da una lezione riguardante il linguaggio SQL:

"vi ricordo che in una struttura unica con un joint abbiamo una singola struttura select from where dove nella from dobbiamo mettere tutte le tabelle"

Come detto in precedenza, non sono presenti segni di punteggiatura, senza i quali non è possibile separare frasi diverse tra loro, e perciò concetti logici diversi, inoltre alcune parole non vengono tradotte in modo corretto, come ad esempio la parola *joint*, che in realtà si riferisce all'operatore SQL *JOIN*. Essendo la traduzione impostata in lingua italiana, questo malfunzionamento della trascrizione avviene soprattuto per parole inglesi, ma non solo. Questo problema rende la ricerca di entità significative più difficile, in quanto sovente nelle materie tecniche, vengono utilizzati molti termini inglesi specifici, perciò non identificarli renderebbe il servizio poco efficiente.
Ricapitolando, in questo contesto, l'obiettivo è fornire un servizio che possa migliorare la reperibilità di contenuti didattici per migliorare la *user experience*. In particolare l'obiettivo è quello di indicizzare dei videocorsi tracciando i segmenti video che trattano di specifici argomenti, sulla base dell'analisi del transcript del video. Come corso di riferimento per le prove durante la realizzazione di questa applicazione, si utilizzerà il corso *Basi di dati* citato in fig. 2.1.

Capitolo 3

Analisi stato dell'arte

L'**estrazione di informazioni** (IE) da un testo è legato al problema della semplificazione del contenuto al fine di creare una vista strutturata delle informazioni presenti in un testo libero. L'obiettivo generale è quello di creare un risultato più facilmente leggibile per elaborare le frasi. In generale si può applicare a diversi domini: articoli di giornale, pagine Web, letteratura scrientifica, annunci economici o di lavoro, cartelle cliniche, ecc.. Il termine *entità nominata* (named entity) identifica un'unità di informazioni come il nome, anche di persone, organizzazioni, location, ed espressioni numeriche che includono tempo, dati, denaro. La ricerca di entità, è stata identificata dai ricercatori,come un ambito chiave per lo sviluppo del Web semantico. Nel 2001, Berners-Lee [43] ha descritto l'evoluzione dei documenti Web leggibili da umani, a un Web di dati in cui le informazioni vengono fornite con una struttura predefinita, più facile da manipolare per i computer. Il Web semantico è un'estensione del Web attuale che aggiunge nuovi dati e metadati ai documenti Web esistenti in modo che gli elaboratori possano integrare e riutilizzare automaticamente i dati attraverso varie applicazioni. Di conseguenza, estrarre entità nominate da un testo e aggiungere contenuti semantici riguardo queste ultime, con supporto di diverse ontologie[1] e basi di conoscenza[2] (knowledge bases, KB), come Wikipedia, hanno sempre attirato di più l'attenzione dei ricercatori. IE fa parte di una visione più grande che affronta il problema di elaborare metodi automatici per la gestione del testo, oltre alla sua trasmissione, memorizzazione e visualizzazione. Un altro approccio complementare è quello dell'**elaborazione del linguaggio naturale** (NLP, *Natural Language Processing*). Quest'ultimo è il processo di trattamento automatico, mediante un calcolatore elettronico, delle informazioni scritte o parlate in una lingua naturale. Questo processo è reso particolarmente difficile e complesso a causa delle caratteristiche intrinseche di ambiguità del linguaggio umano. Per questo motivo il processo di elaborazione viene suddiviso in fasi diverse, tuttavia simili a quelle che si possono incontrare nel processo di elaborazione di un linguaggio di programmazione:

[1]Un'ontologia è una rappresentazione formale di un dominio di interesse che permette di specificare, in modo aperto e significativo, i concetti e le relazioni.

[2]Una base di conoscenza è un tipo speciale di database per la gestione della conoscenza per scopi aziendali, culturali o didattici. Essa costituisce un ambiente volto a facilitare la raccolta, l'organizzazione e la distribuzione della conoscenza. Ogni entità nella base di conoscenza è identificata da una URI (Uniform Resource Identifier), una stringa di caratteri usata per identificare la risorsa.

- analisi lessicale: scomposizione di un'espressione linguistica in token (in questo caso le parole)
- analisi grammaticale: associazione delle parti del discorso a ciascuna parola nel testo
- analisi sintattica: arrangiamento dei token in una struttura sintattica
- analisi semantica: assegnazione di un significato (semantica) alla struttura sintattica e, di conseguenza, all'espressione linguistica

La comprensione del linguaggio naturale è spesso considerata un problema IA-completo, poiché si pensa che il riconoscimento del linguaggio richieda una conoscenza estesa del mondo e una grande capacità di manipolarlo. Per questa ragione, la definizione di *comprensione* è uno dei maggiori problemi dell'elaborazione del linguaggio naturale, dovuta alla forte presenza di parole che possono essere ambigue. Infine un breve riferimento anche al **Text Mining**. Lo scopo di quest'ultimo è elaborare informazioni non strutturate (testuali), estrarre indici numerici significativi dal testo e, quindi, rendere le informazioni contenute nel testo accessibili ai vari algoritmi di data mining (statistica e apprendimento automatico). Le informazioni possono essere estratte per ricavare riepiloghi per le parole contenute nei documenti o per elaborare riepiloghi per i documenti in base alle parole in essi contenute. Quindi, è possibile analizzare parole, gruppi di parole utilizzate nei documenti, ecc. Oppure è possibile analizzare i documenti e determinare le somiglianze tra di essi o il modo in cui sono correlati ad altre variabili di interesse nel progetto di data mining. Nei termini più generali, il text mining *trasforma il testo in numeri* (indici significativi), che possono poi essere incorporati in altre analisi comeprogetti di data mining predittivi , applicazione di metodi di apprendimento non supervisionati (clustering), ecc.

3.1 Named entity recognition (NER)

Identificare riferimenti di queste entità in un corpus di input è un importante obiettivo all'interno dell'ambito di IE e viene chiamato **Named entity recognition** (NER), il quale ha il compito di localizzare e identificare entità nominate all'interno di un testo non strutturato[3], e assegnarle a categorie predefinite. In generale, il problema di NER si può ricondurre alla ricerca di proprietà associate ad una determinata parola. Negli ultimi dieci anni, NER è diventato un argomento molto interessante, che attrae molto sforzo di ricerca, con vari approcci introdotti per diversi domini, ambiti e scopi [5, 40, 7, 9]. Alcuni lavori su NER affrontano il compito di classificazione di NE in ampie categorie [12, 40, 7] e, tra le più importanti, si distinguono:

- **ENAMEX**: si limita ai nomi, acronimi, e altri identificativi univoci, che sono caratterizzati con l'attributo TYPE tramite PERSON (nomi personali, familiari), ORGANIZATION (nomi di corporazioni, governi e altre entità organizzative) e LOCATION (nomi di location politiche o geografiche, come regioni, città, continenti, montagne, laghi ecc.. e località astronomiche).

[3]I dati non strutturati (o informazioni non strutturate) sono informazioni che non hanno un modello di dati predefinito o non sono organizzati in modo predefinito

- **TIMEX**: si riferisce ad espressioni temporali, che può essere caratterizata tramite l'attributo TYPE, in DATE (data completa o parziale), TIME (ora del giorno completa o parziale), DURATION (misura del tempo trascorso durante un periodo noto).

- **LOCATION**: è utile per le espressioni numerali, compresi i valori con le unità di misura. Tramite l'attributo TYPE si caratterizza in MONEY (espressione monetaria), MEASURE (standard per le misurazioni, come età, area, distanze, energia, velocità, temperatura, volume, pressione ecc..), PERCENT (percentuali), CARDINAL (numero o quantità numerica di qualche oggetto).

Altri invece classificano le NE in più categorie più ristrette che sono specificate da una ontologia [5, 9].

3.2 Named entity disambiguation (NED)

Named entity linking (NEL) o **named entity disambiguation** (NED) ha invece il compito di determinare l'identità di entità menzionate all'interno di un testo. Questo è fatto legando le entità con una base di conoscenza. Una delle situazioni più ardue, e comuni, consiste nell'avere una parola che può avere significati diversi, a seconda del contesto in cui viene usata. Ad esempio, se si prende in considerazione la frase: *Paris is the capital of France*, l'idea è quella di determinare che *Paris* fa riferimento alla città di Parigi, e non a Paris Hilton. Questo tipo di ambiguità rende l'identificazione delle entità più complessa. Identificare le entità corrette all'interno del transcript di partenza è essenziale, in particolare l'obiettivo è riuscire ad identificare tutte quelle inerenti al corso in questione. Il raggiungimento di questo fine permetterebbe agli studenti di poter trovare qualsiasi argomento affine al corso, rendendo la *user experience* molto più gradevole. Negli scorsi anni molti approcci sono stati proposti per fare NED [13, 34, 26]. Named entity disambiguation può essere considerato un caso speciale di **Word Sense Disambiguation** (WSD) [27]. L'obiettivo di WSD è quello di identificare il senso di una parola, in base al contesto in cui compare quando esistono diversi significati. Tecniche utilizzate fino ad ora comprendono l'uso di dizionari, dizionari dei sinonimi, o di un'ontologia come inventario di tutti i possibili significati esistenti associati ad una parola. Recentemente, l'enciclopedia più grande e ampiamente usata esistente, Wikipedia, è stata usata come fonte di conoscenza non solo per WSD [25], ma anche per IE, NLP[4], creazione di ontologie, e così via[29].

3.2.1 Basi di conoscenza

In letteratura, le fonti di conoscenza utilizzate per NED possono essere divise in due tipi: *close ontologies* e *open ontologies*. Le ontologie chiuse sono costruite da esperti seguendo un approccio top-down, con una gerarchia di concetti basati su un vocabolario controllato e rigorosi vincoli, ad es. KIM, WordNet. Queste basi di conoscenza sono generalmente di alta affidabilità, ma le loro dimensioni e copertura

[4]L'elaborazione del linguaggio naturale, detta anche NLP, è il processo di trattamento automatico mediante un calcolatore elettronico delle informazioni scritte o parlate in una lingua naturale.

sono limitate. Inoltre, non solo la costruzione di esse è costosa in termini di manodopera, ma non vengono aggiornati delle nuove scoperte e argomenti che sorgono ogni giorno. Nel frattempo, le ontologie aperte sono invece costruite da collaborazioni di volontari seguendo un approccio bottom-up, con concetti formati da una libera comunità, ad es. Wikipedia. Molte ontologie aperte sono in rapida crescita con un'ampia copertura in diversi argomenti e vengono aggiornati quotidianamente. Wikipedia è considerata un'ontologia aperta: gli articoli in essa contenuti si ritengono avere un'alta qualità, c'è però chi ha dei dubbi in merito. Infatti, in [11], Giles ha studiato l'accuratezza di alcuni contenuti di articoli in Wikipedia rispetto a quelli di articoli nell'Enciclopedia Britannica, dimostrando che entrambe le fonti erano ugualmente inclini a errori significativi. Le principali basi di conoscenza pubbliche sono le seguenti:

- **Wikipedia**[5] è un'enciclopedia multilingue online gratuita creata attraverso sforzi collettivi di un numero enorme di volontari intorno al mondo. Al giorno d'oggi, Wikipedia è diventata l'enciclopedia più grande e più popolare al mondo disponibile sul Web ed è inoltre una risorsa molto dinamica e in rapida crescita. Wikipedia è composta da pagine (articoli) che definiscono e descrivono entità o un argomento, e ciascuna di queste pagine fa riferimento a un identificatore univoco. Attualmente, la versione inglese di Wikipedia contiene oltre 5,7 milioni di pagine[6]. Wikipedia ha una vasta copertura di entità e contiene una conoscenza esauriente di entità notabili. Inoltre, la struttura di Wikipedia fornisce un insieme di funzioni utili per il collegamento di entità utilizzando etichette, categorie, pagine di reindirizzamento, pagine di disambiguazione e link ad altre pagine Wikipedia;

- **DBpedia**[7][16] è una base di conoscenza costruita sopra Wikipedia. Viene creato utilizzando informazioni strutturate (infobox, gerarchia delle categorie, coordinate geografiche e collegamenti esterni) contenuti in ogni pagina di Wikipedia. Come Wikipedia, esiste anche in più lingue. Una grande ontologia è usata per modellare i dati e il il numero di entità cresce in modo simile a Wikipedia ad ogni aggiornamento;

- **Wikidata**[8] [10] è un progetto di Wikimedia che vuole essere un punto centrale per i contenuti in arrivo dai diversi progetti Wikimedia. Ha uno schema evolutivo in cui le nuove proprietà, richieste dalla comunità, vengono regolarmente aggiunte e fornisce etichette in molte lingue. Ma ancora più importante, tutte le entità tra le lingue sono collegate e appartengono allo stesso grande grafo. L'obiettivo principale di Wikidata è quello di diventare una base di conoscenza centrale e contiene fino ad ora oltre 25 milioni di entità;

- **YAGO**[9] [8] è una base di conoscenza multilingue che unisce tutte le versioni multilingue di Wikipedia con Wordnet. Usano anche Wikidata per verificare in quale lingua è descritta un'entità. L'obiettivo è fornire una base di conoscenza

[5] `http://www.wikipedia.org`

[6] `http://meta.wikimedia.org/wiki/List\_of\_Wikipedias`

[7] `https://wiki.dbpedia.org/`

[8] `https://www.wikidata.org/wiki/Wikidata:Main\_Page`

[9] `https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/`

per molte lingue che contiene proprietà del mondo reale tra entità e non solo proprietà lessicali;

- **Babelnet**[10] [28] è una base di conoscenza multilingue che unisce Wikipedia, Wordnet, Open Multilingual Wordnet, OmegaWiki, Wikizionario e Wikidata. L'obiettivo è fornire un lessico multilingue e una base di conoscenza semantica che è principalmente basata su relazioni semantiche tra concetti e entità nominate.

3.3 Attività in materia

Molti approcci sono stati proposti per effettuare NED. Tutti si possono raggruppare in tre macro-strategie: **locale, globale e collettivo**. I metodi locali agiscono su ciascuna parola menzionata, in modo indipendente dalle altre, in base alla compatibilità tra il soggetto e le sue entità candidate, utilizzando alcune caratteristiche generali per migliorare la scelta. I metodi globali e collettivi presuppongono che le decisioni di disambiguazione siano interdipendenti e che vi sia coerenza tra entità che occorrono nello stesso testo, consentendo l'uso di misure di relazione semantica per la disambiguazione. I metodi collettivi effettuano simultaneamente le decisioni di disambiguazione, mentre i metodi globali disambiguazione ogni menzione una alla volta.

3.3.1 Approcci locali

Un tipico approccio locale consiste nel valutare la compatibilità tra una menzione e le entità candidate, con il contesto. In primo luogo, le caratteristiche contestuali delle entità all'interno del testo vengono estratte. Successivamente, vengono ponderate e rappresentate in un modello. Infine, ogni menzione nel testo viene collegata all'entità candidata che ha il più alto valore di similarità con essa. Bunescu e Paşca [4] hanno proposto un metodo che usa un kernel SVM per confrontare il contesto intorno alla menzione con quello delle entità candidate, in combinazione con la stima della correlazione della parola con le categorie delle entità candidate. Ogni entità candidata è una entry in Wikipedia e il suo contesto lessicale è il contenuto dell'articolo. Mihalcea e Csomai [34] hanno implementato e valutato due diversi algoritmi di disambiguazione. Il primo basato su una misura di sovrapposizione contestuale tra il contesto della menzione ed i contenuti degli articoli di Wikipedia candidati per identificare il più simile tra l'entità candidate. Il secondo addestra un classificatore di *Naïve Bayes* per ogni menzione ambigua usando tre parole a sinistra e a destra degli outlink negli articoli di Wikipedia. Zhang in [46] utilizza algoritmi di classificazione per imparare compatibilità nel contesto di disambiguazione. Zheng in [51], Dredze in [24] e Zhou in [50] utilizzano tecniche di apprendimento per classificare tutte le entità candidate e collegare la menzione a quella più probabile. Zhang in [45, 47] migliora il suo approccio in [46] con un modello per generare automaticamente un set di training molto grande e addestrare un classificatore statistico per rilevare le varianti di un nome. Han and Sun in [14] hanno proposto un modello probabilistico generativo che combina tre evidenze: la distribuzione delle entità nel documento, la distribuzione dei possibili nomi di una specifica entità, e la distribuzione dei possibili

[10] `https://babelnet.org/`

contesti di una specifica entità. Lo svantaggio principale degli approcci locali è che non tengono conto dell'interdipendenza tra decisioni di disambiguazione.

3.3.2 Approcci globali

Gli approcci globali presuppongono l'interdipendenza tra le decisioni di disambiguazione e sfruttano due principali tipi di informazioni: il contesto di disambiguazione e la relazione semantica. Cucerzan in [6] è stato il primo a usare un modello di interdipendenza tra le decisioni di disambiguazione. In [6] il contesto di disambiguazione è composto da tutti i contesti di Wikipedia che ricorrono nel testo e la relazione semantica si basa sulla sovrapposizione in categorie di entità di cui si potrebbe fare riferimento nel testo. I contesti di Wikipedia comprendono etichette inlink, etichette outlink e i titoli di tutti gli articoli. Milne e Witten [26] hanno proposto un apprendimento basato su un metodo che classifica ciascun candidato in base a tre fattori: la relazione semantica del candidato con le entità contestuali, la comunanza del candidato, definita come il numero di volte che esso è usato come destinazione in Wikipedia, e una misura della qualità generale delle entità contestuali. Guo in [49] ha costruito un grafo unidirezionale G = (E, V), dove V contiene il nome delle menzioni e tutti i loro candidati. Ogni arco si connette da un'entità a una menzione o viceversa; non c'è nessun arco che collega due menzioni o due entità. Quindi l'approccio classifica i candidati di una certa menzione in base al loro ordine di archi entranti ed uscenti. Hachey in [3] costruì prima un grafo G = (E, V) dove V contiene i candidati di tutte le menzioni ambigue. Il grafo è stato successivamente ampliato attraversando, con una lunghezza limitata, i percorsi tramite i link in entrambe le pagine di entità e categoria in Wikipedia, e aggiungendo nodi. Infine, l'approccio classifica le entità candidate. Ratinov in [20] ha proposto un approccio che combina sia l'approccio locale che quello globale estendendo i metodi proposti in [4] e [26]. Kataria in [38] ha proposto un LDA semi-supervisionato per modellare le correlazioni tra le parole e tra gli argomenti per la disambiguazione.

3.3.3 Approcci collettivi

Kulkarni in [39] ha proposto il primo approccio collettivo per effettuare disambiguazione, che può collegare simultaneamente entità menziona in un testo alle corrispondenti voci in una KB ed ha introdotto il problema dell'ottimizzazione a questo fine. L'approccio combina la compatibilità locale tra le menzioni e le loro entità candidate e la relazione semantica tra entità. Poiché l'ottimizzazione è problema arduo da risolvere, essendo *NP-hard*, gli autori hanno proposto dei metodi di approssimazione per risolverlo. Kbleb e Abecker [19] hanno proposto un approccio che sfrutta un grafo RDF, cioè un grafo strutturato. L'approccio applica il metodo *Spreading Activation* per classificare e generare il grafico Steiner più ottimale in base ai valori di attivazione. Il grafo risultante contiene entità di una KB che effettivamente sono referenziate nel testo. Alcuni lavori di ricerca [15, 48] hanno creato un grafo di riferimento per un testo e proposto un metodo di inferenza collettiva per fare *disambiguation*. Un grafo di riferimento è ponderato e non orientato G = (E, V), dove V contiene tutte le menzioni nel testo e tutti i possibili candidati di queste menzioni. Ogni nodo rappresenta una menzione o un'entità. Il grafico ha due tipi di archi:

- Un arco di menzione-entità viene stabilito tra una menzione e un'entità e ponderato sulla base della somiglianza del testo;
- Un arco entità-entità viene stabilito tra due entità e ponderato utilizzando la relazione semantica fra loro.

Han and Sun [48] e Hoffart in [15] proposero approcci che sfruttano la compatibilità del contesto locale e la coerenza tra le entità per costruire un grafo di riferimento e successivamente proposero un approccio collettivo basato sul grafo in combinazione con misure di popolarità di menzioni o entità per l'identificazione simultanea di voci in una KB di tutte le citazioni nel testo. Hoffart in [15] ha proposto un metodo collettivo per fare disambiguazione basata su una stretta ontologia, ontologia YAGO. In seguito, ha calcolato il peso di ogni arco di menzione basato sulla popolarità delle entità e somiglianza testuale, che è composta da similarità *keyphrase-based* e basata sulla sintassi. Han e Sun [48] hanno dapprima costruito un grafico referente dove la compatibilità del contesto locale è stata calcolata in base al modello *bag-of-words* come in [4] e per la relazione semantica è stata adottata la formula presentata in [26].Secondo, gli autori hanno proposto un algoritmo collettivo. L'algoritmo raccoglie le evidenze iniziali per ogni menzione e quindi rafforza queste evidenze propagandole tramite gli archi del grafo di riferimento. L'evidenza iniziale di ogni menzione mostra la sua popolarità sulle altre menzioni e il suo valore è il punteggio TF-IDF[11] normalizzato dalla somma dei punteggi TF-IDF di tutte le menzioni nel testo.

[11]La funzione di peso tf-idf è una funzione utilizzata in *information retrieval* per misurare l'importanza di un termine rispetto ad un documento o ad una collezione di documenti

Capitolo 4

Gerbil e relativi estrattori

Negli ultimi anni, è cresciuto esponenzialmente l'interesse per il riconoscimento di entità nominate all'interno di testi non strurrato. Per questo motivo, sono stati sviluppati molti servizi per l'estrazione di informazioni strutturate da testi pubblicati sul Web. Questi servizi (chiamati estrattori) sono stati resi disponibili sul Web, accessibili tramite specifiche API, rivolte a ricerche pubbliche o per usi commerciali. In questa sezione, verranno analizzati gli estrattori migliori presenti in letteratura, che effettuano disambiguation. Esiste in letteratura uno strumento chiamato GERBIL[1] che è stato sviluppato per disambiguare, usando più estrattori contemporaneamente, dando modo di poter paragonare in modo dettagliato questi strumenti. Analizzare questi servizi presenti nello stato dell'arte è essenziale ai fini del progetto, infatti se fossero sufficienti quest'ultimi per l'analisi dei dati di input (transcript), non sarebbe necessario ideare un nuovo modello. Come si vedrà successivamente, questi estrattori funzionano molto bene per testi inglesi generali, i quali non contengono errori sintattici. Questi strumenti però, *faticano* a dare dei buoni risultati nei seguenti casi:

- non tutti supportano l'italiano;
- quando il contesto non è più generico, ma entra nello specifico (come può essere una qualsiasi lezione di un corso universitario), molte entità appartenenti ad aspetti tecnici specifi, non vengono identificate, dovuto anche al fatto che le basi di conoscenza su cui si basano molte volte, non spiegano i concetti nello specifico;
- utilizzano la punteggiatura per identificare frasi logiche diverse, la quale però non è presente nei nostri transcript;
- essendo presenti parole mal trascritte, può capitare che traducano male il significato di tale menzione, cosa che non dovrebbe capitare riguardo le entità affini al corso.

4.1 GERBIL

La necessità di collegare documenti non strutturati ai dati strutturati, ha portato allo sviluppo di un considerevole numero di annotatori, disponibili online. Tuttavia,

[1] `http://gerbil.aksw.org/`

questi strumenti sono attualmente ancora difficili da confrontare poiché i risultati della valutazione pubblicati sono calcolati su diversi set di dati e valutati in base a diverse misure. **GERBIL** nasce per fornire una struttura di valutazione per l'annotazione di entità semantiche, che fornisce agli sviluppatori, agli utenti finali e ai ricercatori, interfacce facili da usare per la valutazione. GERBIL fornisce una GUI che consente la configurazione e l'eseuzione di esperimenti, assegnando URL persistenti agli esperimenti (migliore riproducibilità e archiviazione), esportando i risultati degli esperimenti in formati leggibili da elaboratori e visualizzazione dei risultati, con le caratteristiche dei set di dati su cui sono stati eseguiti gli esperimenti. GERBIL è un framework open source ed estensibile che consente di valutare gli strumenti usando differenti annotatori su set diversi, potendo scegliere tra 6 diversi tipi di esperimento. La sua architettura e l'interfaccia sono state progettate per consentire la facile integrazione di annotatori tramite i servizi REST, la facile integrazione di set di dati tramite DataHub1, upload di file o integrazione diretta del codice sorgente, l'aggiunta di nuove misure di performance, la fornitura di diagnostica per gli sviluppatori di strumenti e la portabilità di risultati. Per configurare

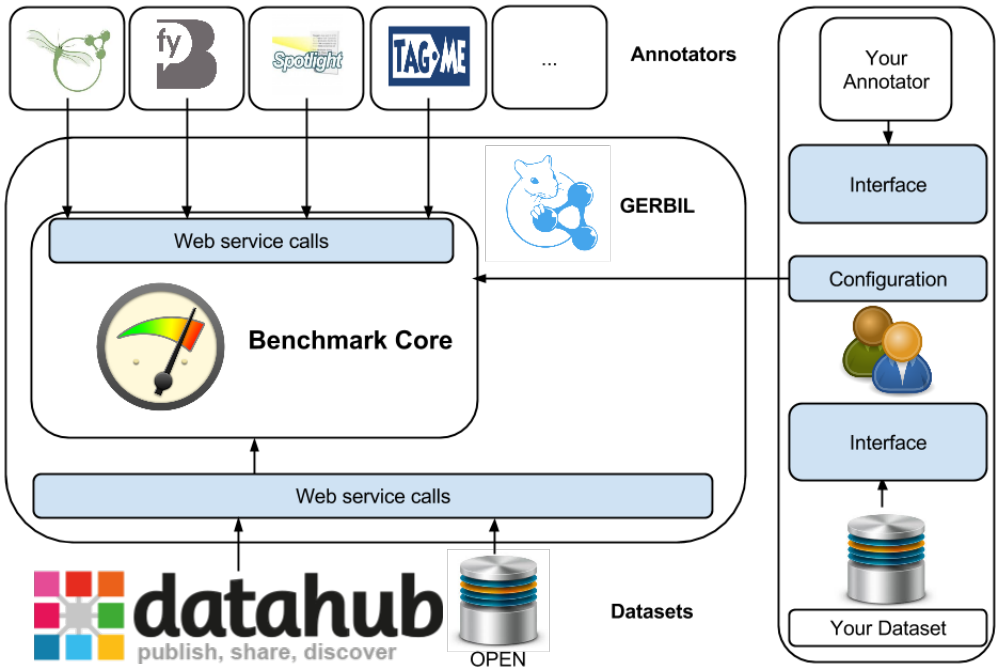


Figura 4.1: Panoramica dell'architettura astratta di GERBIL (fonte: [35]).

un progetto, Gerbil mette a disposizione le seguenti impostazioni:

- il tipo di esperimento. Definisce il modo in cui viene utilizzato per risolvere un certo problema durante l'estrazione di informazioni. GERBIL si estende in diversi tipi di esperimenti [23](compresi *recognition* e *disambiguation*). Con questa estensione, la struttura può trattare con set di dati e annotatori diversi collegate a qualsiasi knowledge base, ad es. DBpedia, BabelNet ecc., a lungo poiché gli identificatori necessari sono gli URI. Esempi di tipi sono *Entity recognition*, *D2KB*, *Entity Typing*, *C2KB*, *A2KB*. In particolare **D2KB** consiste nel mappare delle entità presente in un testo, a quelle presenti in una base di conoscenza. Durante le valutazioni del progetto proposto si utilizzerà questa configurazione;

- Matching. GERBIL offre tre tipi di corrispondenza tra un gold standard e i risultati dei sistemi di annotazione;

- Metrica. Attualmente, GERBIL offre sei misure suddivise in due gruppi: micro e macro gruppo di precisione, recall e f-measure. Questi risultati vengono visualizzati utilizzando diagrammi di spider interattivi che consentono agli utenti di ottenere facilmente una panoramica delle prestazioni dei singoli strumenti e di confrontarli tra loro, raccogliendo informazioni sulle prestazioni.

- Annotatori. L'obiettivo principale è semplificare il confronto di sistemi di annotazione di entità nuove ed esistenti in modo completo e riproducibile. Pertanto, GERBIL offre diversi modi per implementare nuovi framework di annotazione di entità. E' possibile inoltre integrare annotatori usando un adattatore Java, così come un servizio basato su NIF [37]. Attualmente, offre diversi sistemi di annotazione delle entità con una varietà di funzionalità, funzionalità ed esperimenti preconfezionati, inclusi DBpedia Spotlight, TagMe, AIDA, KEA, WAT, AGDISTIS, Babelfy, NERD-ML e Dexter [35].

- Dataset. Sono disponibili diversi insiemi di dati tramite GERBIL. Grazie al gran numero di formati, argomenti e caratteristiche dei set di dati, consente di effettuare diversi esperimenti, dando soprattutto l'opportunità di caricarne altri, a patto che rispettino il formato richiesto.

- Output. L'obiettivo principale di è fornire una visione completa e riproducibile dei risultati degli esperimenti pubblicabili. Quindi, la produzione sperimentale è rappresentata come una tabella contenente i risultati.

Nel seguito verranno presentati in breve alcuni degli estrattori presenti su GERBIL i quali verranno utilizzati per la verifica finale.

4.2 Babelfy

Babelfy[2] è un approccio unificato, multilingue, basato su grafici, per effettuare *Entity Linking* e Word Sense Disambiguation, basandosi su una libera identificazione dei significati dei candidati con una euristica che sfrutta un grafo denso. È posibile leggere i dettagli implementativi a riguardo in [2]. Esso si basa su Babelnet e svolge congiuntamente disambiguazione ed entity linking , in tre fasi:

- Si associa a ciascun vertice della rete semantica BabelNet, vale a dire il concetto o entità nominata, una firma semantica, cioè un insieme di vertici correlati. Questo è un passaggio preliminare che deve essere eseguito solo una volta, indipendentemente dal testo di input;

- Dato un testo di input, estrae tutti i frammenti collegabili da questo testo e, per ognuno di essi, elenca i possibili significati secondo la rete semantica;

[2] `http://babelfy.org/login`

- Crea un'interpretazione semantica basata sul grafico dell'intero testo collegando i significati candidati dei frammenti estratti usando le firme semantiche precedentemente calcolate. Quindi estrae un sottografo denso di questa rappresentazione e seleziona il miglior significato candidato per ciascun frammento.

Babelfy tiene conto di 271 lingue senza assumere o tentare di inferire la lingua del testo di input, dando l'ulteriore possibilità di annotare il testo scritto in più lingue. In fig. 4.2 ho riportato un esempio di risposta fornita dalla demo di Babelfy. Si può

Figura 4.2: Screenshot della demo di Babelfy con un esempio di query e il corrispondente risultato.

notare che è possibile inserire la lingua, ed il risultato è mostrato inizialmente con la lingua originale, ma è possibile trasformarlo in una qualsiasi delle lingue supportate. Infine Babelfy effettua una distinzione tra concetti e entità rilevate nell'input.

4.3 DBpedia Spotlight

DBpedia Spotlight[3]è un progetto open source responsabile dello sviluppo di un sistema per l'annotazione automatica delle entità DBpedia di testo in linguaggio naturale. Fornisce interfacce per la localizzazione di frasi (riconoscimento di frasi da annotare) e disambiguazione (collegamento entità) . In particolare, il loro approccio si articola in tre fasi. La fase di *spotting*, cioè di riconoscimento, individua in una frase le stringhe che possono indicare una menzione di una risorsa DBpedia. La fase di selezione (*candidate selection*) consiste nel cercare tutti i candidati più opportuni per la menzione corrente di cui non è ancora chiaro il significato. La fase di disambiguazione, a sua volta, utilizza il contesto attorno alla menzione per decidere la scelta migliore tra i candidati. L'algoritmo di spotting della frase principale utilizza una corrispondenza esatta delle stringhe, che utilizza l'implementazione *Aho-Corasick*di LingPipe 8[4]. Per quanto riguarda la disambiguazione, l'algoritmo usa la similarità *cosine* e una modifica pesata di *TF-IDF*. L'annotazione può essere personalizzata dagli utenti in base a delle esigenze specifiche attraverso parametri di configurazione (tra i quali fornisce diversi formati di output). In fig. 4.3 si può vedere un esempio di output annotato della demo di DBpedia Spotlight, in inglese

[3]`http://demo.dbpedia-spotlight.org/`
[4]`http://alias-i.com/lingpipe`

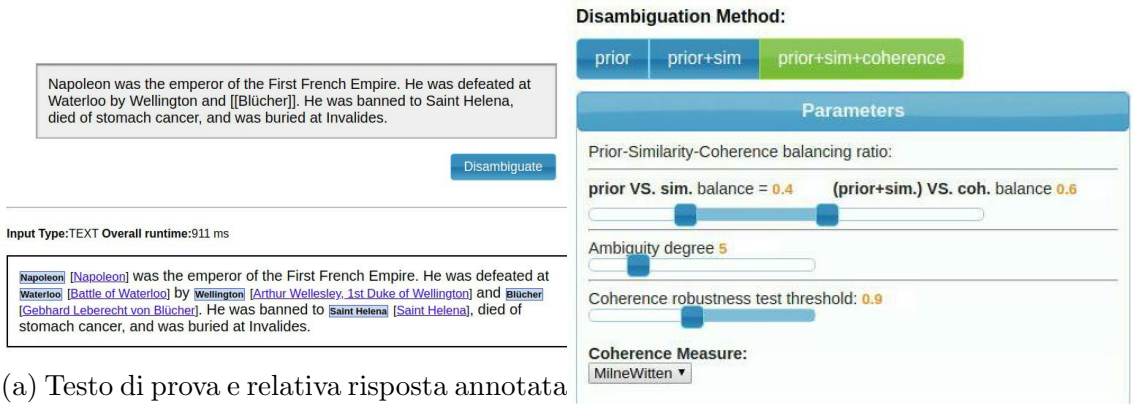
dato che la demo fornisce solo tre lingue (inglese, portoghese e tedesco). È possibile

Figura 4.3: Screenshot della demo di DBpedia Spotlight con un esempio di query e il corrispondente risultato. In particolare viene trovata l'entita SQL corrispondente a *http://dbpedia.org/page/SQL*

leggere una descrizione dettagliata di DBpedia Spotlight in [17] e [31].

4.4 PBOH

PBOH è un approccio alternativo, più veloce ed efficente, rispetto a quelli basati su machine-learning. Questo modello si basa principalmente su semplici statistiche empiriche acquisite da un set di dati di formazione e fa affidamento su un numero molto piccolo numero di parametri appresi. Questo ha alcuni vantaggi, come una procedura di allenamento molto veloce che può essere applicata a enormi quantità di dati, nonché una migliore comprensione del modello rispetto al *deep learn* sempre più popolare. Il punto di partenza utilizza alcune assunzioni naturale che ogni entità possiede, cioè essa dipende dalla sua menzione, dalle sue parole contestuali locali vicine, e dalle altre entità che compaiono nello stesso documento. Per far rispettare queste condizioni, fanno affidamento ad un modello probabilistico condizionale costituito da due parti: la probabilità che un'entità candidata abbia ricevuto il token di riferimento span e il suo contesto circostante, e un conteggio della distribuzione dell'entità candidata corrispondente a tutti le citazioni in un documento. PBOH si basa sull'algoritmo *maxproduct* per dedurre collettivamente le entità per tutte le citazioni in un dato documento. Perciò riassumendo, ogni vincolo di menzione evidenziato è costituito da un'insieme di possibili entità candidate le quali hanno una dimensione limitata, ma lascia un livello significativo di ambiguità. Infine, c'è un modo collettivo di collegamento che è congiuntamente coerente con tutte le entità scelte e supportate dal modello contestuale. È possibile leggere i dettagli a riguardo in [30].



(a) Testo di prova e relativa risposta annotata

(b) Parametri di configurazione

Figura 4.4: Esempio di query effettuata sul sistema AIDA

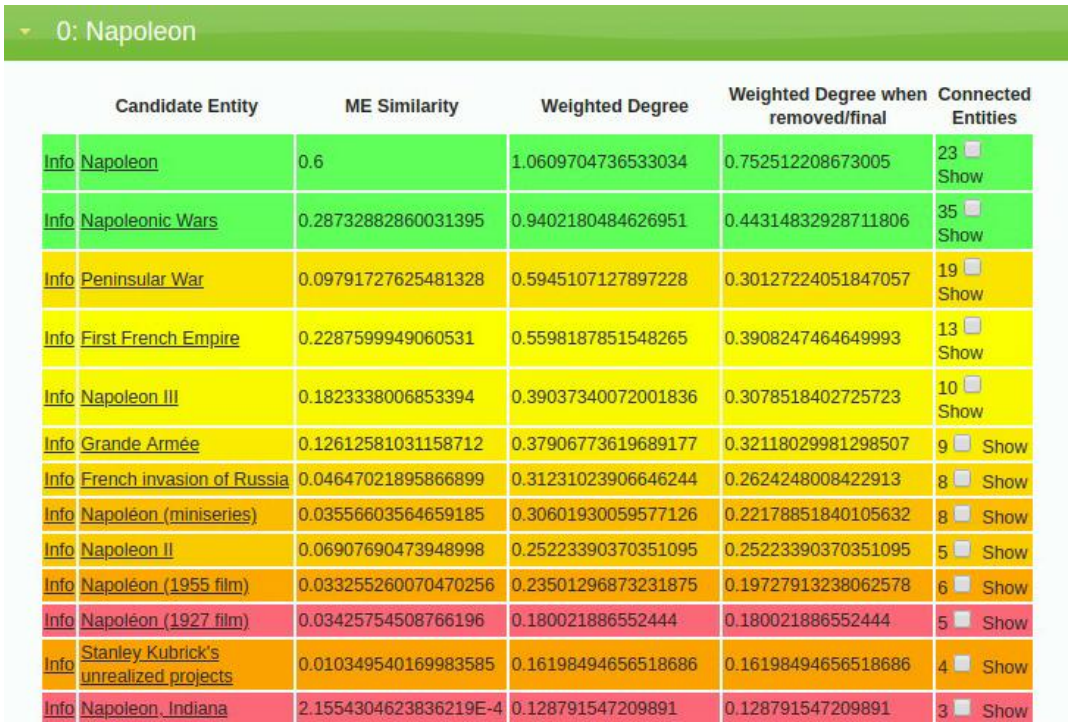
4.5 AIDA

AIDA[5] nasce con l'idea che Wikipedia in inglese è adatta solo per disambiguare testi in inglese generale come articoli di notizie in inglese. Sviluppare sistemi di disambiguazione per altri domini e linguaggi richiede un grande adattamento per adattarsi agli scenari di applicazione specifici. Inoltre, le edizioni di Wikipedia di molte lingue, come l'arabo, sono di un ordine di grandezza inferiore alla Wikipedia inglese. Perciò, è fondamentale sfruttare i collegamenti inter-lingue per arricchire le risorse non inglesi. AIDA sfrutta la base di conoscenze YAGO2, come catalogo di entità e una ricca fonte di relazioni tra entità. Il sistema disambigua sfruttando un grafo: menzioni dal testo di input e le entità candidate definiscono il set di nodi, e gli archi tra menzioni ed entità, vengono ponderati catturando somiglianze di contesto, mentre archi tra entità, ponderati utilizzando una misura di coerenza. I dettagli implementativi sono ampiamente spiegati dagli autori in [18].

Nella figura 4.4 è riportata una demo di questo sistema. In particolare:

- L'interfaccia utente online offre tre metodi principali: *prior only*,*prior* e *similarity* insieme, e il metodo *graph-based* che integra *prior*,*similarity* e *coherence*;
- L'utente può inserire qualsiasi testo, anche in formato HTML. Di default, lo Stanford NER Tagger identifica frasi nominali che possono essere interpretate come entità-menzioni. Poiché questo è soggetto a errori, l'utente può alternativamente segnalare menzioni specifiche mettendole tra doppie parentesi;
- L'output mostra per ogni menzione (in verde), l'entità che il metodo scelto ha assegnato alla menzione, sotto forma di link cliccabile. I collegamenti puntano alle corrispondente pagine in Wikipedia.
- Per ogni testo inserito, una volta disambiguato, l'utente può vedere diverse informazioni e statistiche sul tempo di esecuzione relative all'input, il grafo costruito, ecc.

[5] `https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/ambiverse-nlu/aida/`



0: Napoleon

	Candidate Entity	ME Similarity	Weighted Degree	Weighted Degree when removed/final	Connected Entities
Info	Napoleon	0.6	1.0609704736533034	0.752512208673005	23 Show
Info	Napoleonic Wars	0.28732882860031395	0.9402180484626951	0.44314832928711806	35 Show
Info	Peninsular War	0.09791727625481328	0.5945107127897228	0.30127224051847057	19 Show
Info	First French Empire	0.2287599949060531	0.5598187851548265	0.3908247464649993	13 Show
Info	Napoleon III	0.1823338006853394	0.39037340072001836	0.3078518402725723	10 Show
Info	Grande Armée	0.12612581031158712	0.37906773619689177	0.32118029981298507	9 Show
Info	French invasion of Russia	0.04647021895866899	0.3123102390664624	0.2624248008422913	8 Show
Info	Napoléon (miniseries)	0.03556603564659185	0.30601930059577126	0.22178851840105632	8 Show
Info	Napoleon II	0.06907690473948998	0.25223390370351095	0.25223390370351095	5 Show
Info	Napoléon (1955 film)	0.033255260070470256	0.23501296873231875	0.19727913238062578	6 Show
Info	Napoléon (1927 film)	0.03425754508766196	0.180021886552444	0.180021886552444	5 Show
Info	Stanley Kubrick's unrealized projects	0.010349540169983585	0.16198494656518686	0.16198494656518686	4 Show
Info	Napoleon, Indiana	2.1554304623836219E-4	0.128791547209891	0.128791547209891	3 Show

Figura 4.5: Lista di candidati per la prima menzione trovata nella query in fig.4.4, ordinate per importanza cronologica

- Per ogni menzione, si può ispezionare le entità candidate e i pesi associati per somiglianza e coerenza. Le entità sono ordinati nell'ordine in cui sono stati eliminati dal set dei candidati (rimosso dal grafo). La colorazione riflette questo ordine: l'entità finale selezionata è mostrata in verde e le alternative sono giallo-rosse. La tonalità del colore indica il range temporale in cui il candidato è stato escluso, cioè tanto più è scuro il colore, quanto più indica che il candidato è stato scartato presto. Un esempio si può vedere in fig. 4.5, la quale mostra le prime righe della prima menzione trovate nella query eseguita in fig. 4.4.

4.6 AGDISTIS

Il framework di AGDISTIS[6] si rivolge a due dei principali inconvenienti dei framework di linking delle entità correnti: complessità temporale e accuratezza. AGDISTIS, è un framework che raggiunge la complessità del tempo polinomiale Il framework è *knowledge-base-agnostic* (cioè, può essere implementato su qualsiasi base di conoscenza) ed è anche indipendente dalla lingua. Nella demo presente online, AGDISTIS è distribuito su tre lingue diverse (inglese, tedesco e Cinese) e tre diverse basi di conoscenza (DBpedia, DBpedia tedesco e DBpedia cinese). La versione online della demo è disponibile su `http://agdistis.aksw.org/demo` ed è rappresentata in 4.6. Dopo aver digitato o incollato il testo nel campo di input, gli utenti possono scegliere tra annotare le entità manualmente o avere le entità rilevato automaticamente. Nel primo caso, le etichette delle entità devono essere segnato usando parentesi quadre (come mostrato in 4.6). Una volta che l'utente ha impostato quali

[6] `http://aksw.org/Projects/AGDISTIS`

Figura 4.6: Screenshot della demo di AGDISTIS con un esempio inglese già annotato.

entità devono essere disambiguate, il testo marcato viene inviato al modulo di rilevamento della lingua basato su [41]. Questa libreria perché è sia precisa (precisione maggiore del 99%) ed efficiente nel tempo. Il vantaggio principale di questo approccio è che l'utente non ha bisogno per selezionare la lingua in cui il testo è esplicitato manualmente, portando così a un'esperienza utente migliorata. Per quanto riguarda la parte di linking, il testo annotato viene inoltrato alla distribuzione specifica della lingua corrispondente di AGDISTIS, di cui ciascuno fa affidamento su una versione specifica per lingua di DBpedia 3.9. L'approccio alla base di AGDISTIS [36] è indipendente dalla lingua e si combina una ricerca in ampiezza con l'algoritmo *HITS*. Inoltre, le misure di similarità delle stringhe e l'euristica di espansione dell'etichetta sono utilizzate per giustificare errori di battitura e variazioni morfologiche nella denominazione.

4.7 FOX

FOX[7] è un framework che fa uso di diversi tipi di algoritmi di NLP per estrarre le triple RDF. FOX integra framework di riconoscimento di entità all'avanguardia utilizzando un apprendimento di tipo ensemble (ensemble learning, EL). La pipeline di FOX è costituito da quattro fasi principali: preelaborazione dei dati di input non strutturati, riconoscimento di Entità nominate, collegamento delle NE riconosciute a risorse che utilizzano AGDISTIS e conversione dei risultati in formato RDF. In fig. 4.7, ho riportato un esempio di utilizzo della demo di FOX, con il corrispondente risultato riportato al fondo della figura. La preelaborazione di FOX consente agli utenti di utilizzare una URL, testo con tag HTML o testo semplice come dati di input (vedi fig. 4.7). L'input può essere eseguito in un form o tramite il servizio web

[7]FOX online demo: `http://fox-demo.aksw.org`, FOX progetto: `http://fox.aksw.org`.

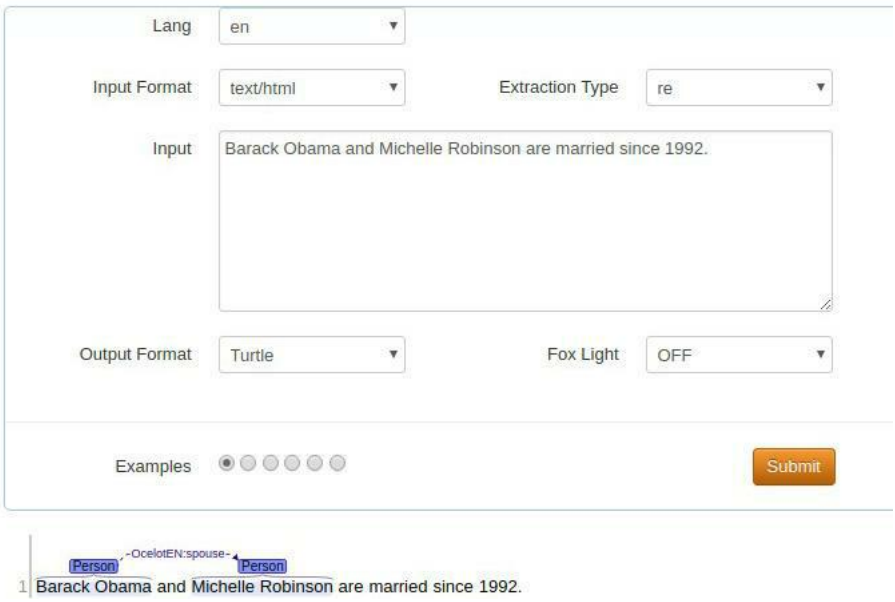


Figura 4.7: Esempio di query e corrispondente risultato della demo di FOX.

di FOX. In caso di URL, FOX invia una richiesta all'URL specificato per ricevere i dati di input. Quindi, per tutti i formati di input, FOX rimuove i tag HTML e rileva frasi e token. Per il riconoscimento di entità si basa su quattro strumenti NER all'avanguardia finora: lo *Stanford Named Entity Recognizer* (Stanford),*Illinois Tagger* (Illinois), *Ottawa Baseline Information Extraction* (Balie) e *Apache OpenNLP Name Finder* (OpenNLP). E' possibile impostare lo strumento preferito tra quelli citati, agendo sul campo *FOX light*. Se quet'ultimo non èstato impostato, FOX utilizza questi quattro strumenti NER in parallelo e memorizza le NE ricevuti per ulteriori elaborazione. Infine, i risultati di tutti gli strumenti sono uniti usando il layer EL di FOX come discusso in [42]. Per la fase di linking, FOX fa uso di AGDISTIS, in grado di collegare le entità a tutte le basi di conoscenza di dati collegati, disambiguare le entità e collegarle a DBpedia, come spiegato in 4.6. Infine per quanto riguarda il formato di output dei dati, fornisce diversi formati: JSON-LD, N-Triples, RDF/JSON, RDF/XML, Turtle, TriG, N-Quads.

Capitolo 5

Metodologia proposta

Questo capitolo sarà focalizzato sulla spiegazione dei passi seguiti per costrutire un prototipo di estrattore. In generale, l'obiettivo è quello di generare un sistema che sia in grado di fare NED sia per contenuti inglesi, sia italiani, dato che l'applicazione è rivolta in primo luogo alle videolezioni in italiano del Politecnico di Torino, e soprattutto di riuscire ad identificare il maggior numero di entità possibili inerenti al corso in questione. Questa ipotesi iniziale infatti è differente dagli altri disambiguatori, i quali cercano di individuare il maggior numero di entità corrette in un contesto generale. Inoltre, a differenza degli estrattori presenti in letteratura, si è dovuto prendere in considerazione delle problematiche aggiuntive, cioè il fatto di dover disambiguare un testo con numerosi errori semantici, e in mancanza di punteggiatura. In questo capitolo si farà riferimento in particolare al corso video registrato di *Basi di dati*, all'interno del secondo anno di Ingegneria Gestionale al Politecnico di Torino, ma i passi seguiti potranno essere applicati ad un qualsiasi corpus di input.

5.1 Schema generale

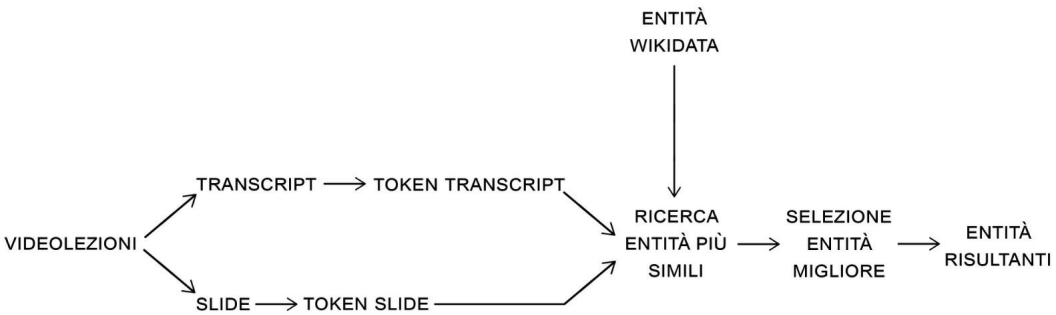


Figura 5.1: Schema riassuntivo delle fasi per la ricerca delle entità

La fig. 5.1 rappresenta una schema ad alto livello delle fasi seguite nella generazione di un nuovo modello di estrattore. Come prima operazione, si è preparata la base di conoscenza, cioè Wikidata, prelevando tutte le entità, con le relative caratteristiche (stringhe che la descrivono), dall'ultimo dump presente online; successivamente, questi dati sono stati salvati in locale, per potervi accedere agevolmente in qualsiasi momento (questa parte corrisponde a *entità wikidata* presente in fig.5.1).

Per quanto riguarda le videolezioni invece, partendo da una generica di queste, è possibile estrarre due tipi di informazione: il contenuto delle slide, quelle utilizzate dal professore nella sua spiegazione, e il *transcript* del parlato del professore. Avendo a disposizione questi due input, viene effettuata una *tokenizzazione*, cioè si identificano le singole parole al suo interno, considerando i segni di punteggiatura a parte e senza considerare gli spazi tra le parole. In seguito vengono generate da questi token tutte le possibili stringhe di riferimento di token ordinati, da 1 a 10 token; nel capitolo 5.4 è presente un esempio che chiarisce questo passaggio. Successivamente, si esegue un'operazione di *confronto* di queste stringhe di riferimento all'interno delle caratteristiche, salvate precedentemente, appartenenti alle entità di Wikidata, salvando tutte le coppie che superano una certa soglia di *similarità* (cioè sono sintatticamente simili e vicine tra loro). A questo punto, ad ogni stringa di riferimento di partenza saranno associate da 0 a n stringhe candidate appartenenti a determinate entità. L'ultima fase (*selezione entità migliore*) consiste nel scegliere tra le entità candidate per una determinata menzione, quella migliore, cioè quella che rappresenta al meglio il significato della menzione all'interno del contesto. Questa è l'idea generale che si è utilizzata nella realizzazione del tool finale all'interno del progetto.
Ricapitolando, si utilizzerà un approccio locale nella scelta dei candidati basato su una similarità tra stringhe, utilizzando Wikidata. Per la fase di *disambiguation*, si seguirà un approccio globale considerando anche il contesto attorno alla menzione nella scelta dell'entità migliore.

5.2 Scenario

Partendo da una generale videolezione di una qualsiasi materia, si hanno a disposizione due tipologie di informazioni: quelle provenienti dalla voce del professore, e quelle provenienti dalle slide, o contenuti scritti sul momento, proiettati a video. Per ottenere il transcript[1] delle parole pronunciate dalla professoressa nel corso, si è utilizzato un servizio offerto da YouTube, il quale, partendo da un generico video, genera un file WebVTT (Web Video Text Tracks) con estensione *vtt*[2]. Questi file sono caratterizati da un elenco di parole e, ad ognuna di esse, è associato un tempo di inizio e di fine con precisione al millisecondo, corrispondente al momento in cui la parola è stata pronunciata. Analizzando i file, è stato possibile notare che i tempi finali non sono affidabili e precisi, perciò per ora, sono stati utilizzati solo i tempi iniziali. Per quanto riguarda le slide nel video, per le elaborazioni si è fatto ampio uso di *OpenCV*[3] eseguendo una serie di operazioni:

- Estrazione di un frame al secondo;
- Rielaborazione dei frame estratti. Per ognuno di essi, è stata estratta solo la parte dell'immagine corrispondente al foglio utilizzato dal professore per scrivere, o alla slide proiettata (in generale infatti, un video è composto da un riquadro in cui si vede fisicamente il professore, ed un riquadro più grande

[1] Il transcript, o trascrizione, in questo ambito corrisponde alle parole dette dal professore, riportate su un file WebVTT.

[2] https://en.wikipedia.org/wiki/WebVTT

[3] OpenCV è una libreria software multipiattaforma nell'ambito della visione artificiale in tempo reale; è disponibile per più linguaggi di programmazione.

corrisponde alla slide, foglio elettronico o ripresa di un foglio normale su cui il professore sta spiegando);

- Ricerca di cambiamento di immagine, corrispondente ad un cambio slide. Partendo dalle immagini rielaborate al punto precedente, utilizzando delle librerie per il processamento di immagini, si è calcolato un indice di similarità strutturale per ognuna di esse. Utilizzando questo valore, è possibile trovare le immagini consecutive diverse, al superamento di un valore di soglia, e di conseguenza, slide diverse;

- Estrazione testo. Per ogni immagine campione in ogni cambio slide rilevato (è sufficiente una sola immagini dato che in qual range di frame sono teoricamente tutte uguali), si è estratto il testo contenuto all'interno della slide, utilizzando una libreria riconoscimento ottico dei caratteri (OCR), la quale riconoscerà e *leggerà* il testo incorporato nelle immagini.

Per quanto riguarda le slide, queste fasi sono necessarie in quanto non è possibile supporre che ogni professore di ogni corso per il quale sarà rivolto questo servizio, metta a disposizione i lucidi utilizzati per l'elaborazione. Inoltre questo metodo funziona solo per contenuti digitali, perciò in corsi dove non si fa utilizzo di slide (ma è il professore stesso a scrivere), le uniche informazioni saranno fornite dal transcript. Ottenuti sia il transcript che il contenuto delle slide, sono stati riformattati in modo da avere per ogni lezione, un file contenente il parlato, ed uno col contenuto delle slide. Come detto nel capitolo introduttivo, questi testi non sono perfetti: il transcript di YouTube, commette molti errori sintattici, traducendo parole simili alla realtà, ma in genere con significati diversi, ed è inoltre privo di segni di punteggiatura, rendendo impossibile il riconoscimento preciso di concetti o frasi; per quanto riguarda il contenuto delle slide, il traduttore funziona per lo più bene, ma solitamente il contenuto di esse è limitato (solitamente il professore preferisce lui stesso scrivere o spiegare i contenuti oralmente).

5.3 Wikidata

Come base di conoscenza di partenza per la creazione dell'estrattore, si è scelto di utilizzare Wikidata[4], a differenza di molti progetti presenti in letteratura che utilizzavano i contenuti presenti negli articoli di Wikipedia. Wikidata è il più giovane dei progetti gestiti dalla Wikimedia Foundation,la fondazione non-profit di diritto statunitense che gestisce,fra gli altri,l'enciclopedia libera Wikipedia. Lo scopo principale di Wikidata è raccogliere e strutturare i dati fondamentali delle voci e delle pagine degli oltre 800 progetti Wikimedia, in modo tale che possano essere letti,tradotti,modificati e riutilizzatida chiunque (macchine comprese) in ciascuna delle 285 lingue ufficiali dei progetti Wikimedia. Wikidata è una base di conoscenza secondaria, libera, collaborativa e multilingua. Più nello specifico:

- secondaria. Wikidata contiene, oltre ai dati strutturati a cui si è accennato in formato 'proprietà : valore', anche le fonti da cui essi vengono tratti, in modo da riflettere la diversità delle stesse nella descrizione di un determinato

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

argomento e ribadire la centralità che ha la nozione di verificabilità in tutti i progetti Wikimedia;

- libera. Tutti i dati sono pubblicati con licenza Creative commons zero (CC0), ossia rilasciati in pubblico dominio, in modo da garantirne il più ampio riuso possibile. In particolare, ciò significa che chiunque può liberamente copiare, modificare, distribuire e utilizzare in ogni forma i dati di Wikidata, anche per uso commerciale, con l'unico obbligo di citare la fonte da cui sono stati prelevati,esattamente come accade in tutti gli altri progetti Wikimedia;

- multilingua. La navigazione, la fruizione, la modifica e il riuso dei dati è garantito nelle 285 lingue ufficiali riconosciute dalla Wikimedia Foundation, oltre che in formati *machine-readable*. L'inglese ha un ruolo prevalente nelle comunicazioni fra membri, ma nell'ambito dati è fortemente incoraggiato l'uso di qualsiasi lingua;

- collaborativa. L'inserimento e la gestione dei dati è supervisionato dalla comunità di Wikidata, che decide inoltre sulle linee guida generali del progetto e su quelle specifiche sui contenuti. In particolare,la comunità gestisce l'importazione di materiale proveniente da fonti affidabili attraverso dei programmi automatici (chiamati bot).

Inoltre si pone fondamentalmente quattro obiettivi: centralizzazione dei link tra pagine, raccolta di dati strutturati (in un formato *machine-readable*), creazione di query automatiche (attualmente è presente un endpoint SPARQL[5]), fornire supporto a parti terze.

5.3.1 Struttura dei dati

Wikidata si divide in vari namespace, ossia in diversi gruppi di pagine che hanno ciascuno una propria funzione. I due più rilevanti sono: quello principale e quello riguardante le proprietà. Il namespace principale (chiamato anche namespace zero o ns0) è l'area del sito in cui sono contenuti gli elementi (item), ossia le pagine che contengono i dati strutturati, provenienti dalle pagine di Wikipedia e degli altri progetti, su un singolo oggetto, soggetto o concetto. Ciascun elemento, inoltre,raggruppa anche tutti i sitelink verso tutti i progetti che contengono una voce o una pagina riguardante quel dato argomento. Tutti gli elementi sono caratterizzati dalla lettera Q seguita da un numero progressivo[6] e sono composti di cinque parti:

- una etichetta (per esempio, SQL);

- una descrizione (per esempio, *language in a computer designed for the retrieval and management of data in relational database management systems as well as database schema and access control management* è la descrizione corrispondente all'entità SQL)

- uno o più alias, ossia nomi alternativi (per esempio, *Structured Query Language*);

[5]https://query.wikidata.org/

[6]Per esempio, l'elemento dedicato a SQL è disponibile all'indirizzo `https://www.wikidata.org/wiki/Q47607`

- un certo numero di sitelink;
- un certo numero di dichiarazioni.

Una dichiarazione è una parte dei dati riguardanti un elemento contenuti in una pagina. Essa consiste in un'affermazione, corredata da riferimenti(ossia le fonti da cui i dati provengono) e da una valutazione (usata per distinguere diverse affermazioni contenute nella stessa proprietà). Un qualificatore è una affermazione che è da

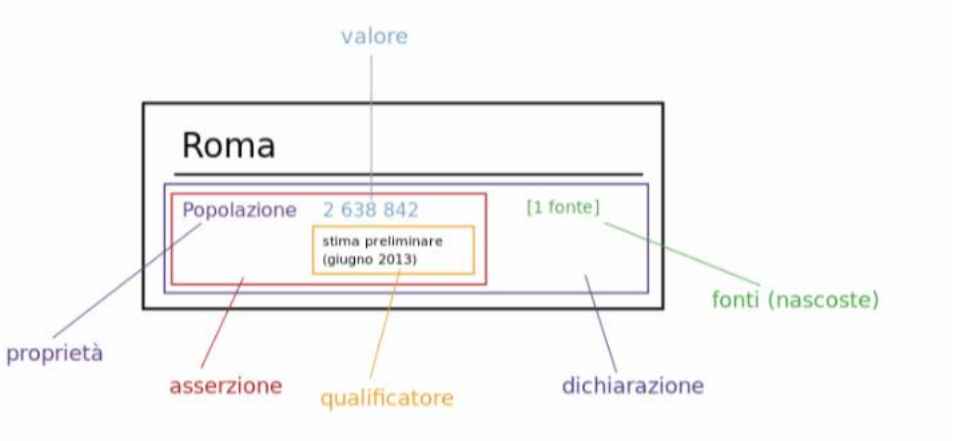


Figura 5.2: Un esempio commentato di dichiarazione.

ritenersi subordinata all'affermazione principale. Una proprietà è un descrittore dei valori, di una relazione, di caratteristiche composite o di un valore eventualmente mancante. Ogni dichiarazione in una pagina di un elemento ha la connessione a una proprietà e le assegna uno o più valori, relazioni, valori compositi o, eventualmente, valori mancanti. Tutte le proprietà sono caratterizzate dalla lettera P seguita da un numero progressivo[7] e sono composte in maniera molto simile agli elementi, possedendo anche loro un'etichetta, una descrizione, uno o più alias e una o più dichiarazioni descrittive della proprietà.

5.3.2 Dump di wikidata

Avendo scelto di utilizzare wikidata come base di conoscenza, si è utilizzata l'ultima versione[8], ed in seguito è stata oppurtunamente filtrata. Come detto in precedenza, la rappresentazione di wikipedia segue il formato RDF in triplette:

SOGGETTO PREDICATO OGGETTO

In dettaglio:

- Soggetto: corrisponde alla URI[9] di una entità, tramite la quale è possibile accedere online alla pagina corrispondente di quest'ultima.
- Predicato: corrisponde ad una proprietà di wikidata.

[7]Per esempio, la proprietà 'istanza di'è disponibile all'indirizzo `https://www.wikidata.org/wiki/P31`

[8]`https://www.wikidata.org/wiki/Wikidata:Database_download/it`

[9]Uniform Resource Identifier (in acronimo URI) è una sequenza di caratteri che identifica univocamente una risorsa generica

- Oggetto: è il valore corrispondente alla proprietà specificata dal campo predicato, corrispondente all'entitità specificata dal soggetto.

```
http://schema.org/description@en: description@en
http://schema.org/description@it: description@it
http://www.w3.org/2000/01/rdf-schema#label@en: label@en
http://www.w3.org/2000/01/rdf-schema#label@it: label@it
http://www.w3.org/2004/02/skos/core#altLabel@en: altLabel@en
http://www.w3.org/2004/02/skos/core#altLabel@it: altLabel@it
http://www.w3.org/2004/02/skos/core#prefLabel@en: prefLabel@en
http://www.w3.org/2004/02/skos/core#prefLabel@it: prefLabel@it
http://www.wikidata.org/prop/direct/P279: subclass_of
http://www.wikidata.org/prop/direct/P31: instance_of
http://www.wikidata.org/prop/direct/P361: part_of
```

Figura 5.3: Proprietà e label di wikidata prese in considerazione dal dump

Esistono attualmente più di duemila proprietà, ma dopo la fase di filtraggio, le uniche che si terranno in considerazone ai fini del progetto, sono tre (rappresentate nelle ultime tre righe in 5.3) corrispondenti a *istance_of*, *subclass_of* e *part_of*. Oltre a queste proprietà principali, il file conterrà anche alcune informazioni aggiuntive, sempre nello stesso formato, rappresentate nelle restanti linee in 5.3. Questi predicati sono:

- la label (o etichetta), cioè il nome più comune con il quale l' oggetto è conosciuto. Non è necessario che sia univoco, in quanto più entità possono avere la stessa etichetta, tuttavia non è possibile che due diverse entità abbiano la stessa etichetta e la stessa descrizione; corrisponde al titolo della pagina in wikidata;

- la descrizione dell'entità (description@en/description@it);

- la prefLabel e altLabel, cioè le label preferite ed alternative dell'entità.

In 5.3, la struttura è composta da due colonne: la prima mostra le URI corrispondenti agli elementi scelti, la seconda il corrispondente nome della tabella in cui verranno salvati in locale . Da notare però che non tutte le entità possiedono tutti questi elementi, perciò i dati reali di partenza conterranno solo i valori effettivamete presenti in Wikidata. La figura 5.4 rappresenta le prime righe del file appena descritto.

```
<http://www.wikidata.org/entity/Q31> <http://www.w3.org/2000/01/rdf-schema#label> "Belgium"@en .
<http://www.wikidata.org/entity/Q31> <http://www.w3.org/2004/02/skos/core#prefLabel> "Belgium"@en .
<http://www.wikidata.org/entity/Q31> <http://www.w3.org/2000/01/rdf-schema#label> "Belgio"@it .
<http://www.wikidata.org/entity/Q31> <http://www.w3.org/2004/02/skos/core#prefLabel> "Belgio"@it .
<http://www.wikidata.org/entity/Q31> <http://schema.org/description> "federal constitutional monarchy in Western Europe"@en
<http://www.wikidata.org/entity/Q31> <http://www.w3.org/2004/02/skos/core#altLabel> "Kingdom of Belgium"@en .
<http://www.wikidata.org/entity/Q31> <http://www.w3.org/2004/02/skos/core#altLabel> "be"@en .
<http://www.wikidata.org/entity/Q31> <http://www.w3.org/2004/02/skos/core#altLabel> "\U0001F1E7\U0001F1EA"@en .
<http://www.wikidata.org/entity/Q31> <http://www.wikidata.org/prop/direct/P31> <http://www.wikidata.org/entity/Q3624078> .
```

Figura 5.4: Esempio contenuto del file contente le triplette del formato RDF provenienti da Wikidata

5.3.3 Salvataggio dati in locale

Nella ricerca di entità all'interno del testo, molte informazioni sono utili per effettuare diversi tipi di filtraggio per ottenere solo i risultati migliori. Per questo motivo, è stato opportuno salvare diversi tipi di dato in un database locale per evitare di dover fare continuamente richieste online all'endpoint SPARQL di Wikidata, operazione che fallisce se le richieste sono troppo numerose o ci sono problemi di connettività. Perciò, nella scelta del database più idoneo per la situazione attuale, si sono valutati alcuni aspetti di database relazionali[10] e database NoSQL[11]. In particolare, i primi hanno i seguenti punti negativi:

- Necessità di definire uno schema prima dell'inserimento dei dati. Alcuni casi d'uso impongono il bisogno di apportare modifiche allo schema (aggiungere o rimodellare dati)e questo processo richiede una significativa quantità di tempo durante il quale il database non può essere utilizzato(downtime);

- Nel modello relazionale i dati devono essere normalizzati per eliminare ridondanze ed anomalie durante gli aggiornamenti. I dati sono suddivisi su più tabelle, ne consegue che le operazioni ed in particolare il JOIN, la più comune nonché la più onerosa in termini di risorse, ha prestazioni che degradano soprattutto con l'aumentare dei dati;

- Object-relational impedance mismatch. Le operazioni SQL non sono adatte per le strutture dati object oriented utilizzate nella maggior parte delle applicazioni odierne;

- Difficoltà a scalare orizzontalmente, cioè aggiungere nodi al pool di risorse e partizionare le tabelle in modo che ogni nodo contenga solo una parte di esse. Inoltre un'altra causa è il fatto che devono garantire le proprietà ACID (atomicità, consistenza, isolamento, durabilità) e supportare le transazioni.

In particolare per il primo e quarto punto, la scelta più logica è stata qualla di scartare un database relazione, e virare verso uno NoSQL. In particolare, i database NoSQL consentono l'inserimento e la manipolazione dei dati senza schema predefinito (di conseguenza è molto facile apportare modifiche significative allo schema dei dati in tempo reale, senza preoccuparsi di possibili interruzioni di servizio). Infine, supportano in modo nativo lo sharding, cioè la capacità di suddividere i dati tra un numero arbitrario di server in maniera del tutto trasparente alle applicazioni, ed hanno elevate prestazioni per operazioni di lettura/scrittura sull'intero data-set (molti sistemi vantano lockless-design che permettono di leggere e scrivere a velocità costante indipendentemente dal numero di utenti che stanno effettuando richieste sui dati). Come database NoSQL si è scelto MongoDB[12]. Quest'ultimo è un DBMS non relazionale, orientato ai documenti, scalabile, ad alte prestazioni. MongoDB si allontana dalla struttura tradizionale basata su tabelle dei database relazionali in

[10]Il modello relazionale è un modello logico di rappresentazione o strutturazione dei dati di un database implementato su sistemi di gestione di basi di dati (DBMS), detti perciò sistemi di gestione di basi di dati relazionali (RDBMS).

[11]NoSQL è un movimento che promuove sistemi software dove la persistenza dei dati è caratterizzata dal fatto di non utilizzare il modello relazionale, usato dalle basi di dati tradizionali.

[12]`https://www.mongodb.com/it`

favore di documenti in stile JSON con schema dinamico (MongoDB chiama il formato BSON), rendendo l'integrazione di dati di alcuni tipi di applicazioni più facile e veloce. Questi documenti vengono salvati in collezioni (corrispondenti alle tabelle in un database relazione). All'interno della stessa collezione, i documenti possono avere formato differente. La struttura utilizzata è rappresentata in 5.3. Per ogni entry nel file vengono inseriti due valori. Nella collezione principale, cioè *wikidata_URI*: una entry in cui la chiave è il soggetto e il valore sono coppie corrispondenti a proprietà-oggetto (dato che la stessa entità compare più volte nel file, ogni chiave corrisponde a più coppie). Il secondo valore inserito corrisponde ad una entry nella

Figura 5.5: Esempio di entry nella collezione principale in mongoDB

Figura 5.6: Esempio di entry nella collezione altLabel in mongoDB

collection corrispondente al nome della proprietà (da cui prende il nome) in cui la chiave è il soggetto, mentre il valore è l'identificativo del soggetto. Ovviamente *wikidata_URI* conterrà la maggior parte dei dati. La figura 5.5 rappresenta una entry nella collezione principale, e come si può vedere dall'esempio, è possibile che alcuni valori non siano presenti, come in questo caso il tipo, oppure che per lo stesso campo, esistano diversi valori. In fig. 5.6 è rappresentata un esempio di entry presente nella tabella altLabel versione italiana: anche in questo caso i valori che assume il campo

wikidata_URI, potrebbero essere molteplici, nel caso in cui la stessa label venisse usata in entità differenti.

5.3.4 Valori aggiuntivi

Come si vede dalla fig. 5.5, sono presenti alcuni campi aggiunti che non sono ancora stati menzionati, i quali saranno analizzati in questo sotto-capitolo. Questi valori sono: il tipo di entità, la distanza dell'entità corrente da Wikidata e da Wikipedia dell'entità principale. Per ottenere la prima informazione, si è utilizzata la seguente query SPARQL:

```
SELECT ?wd_uri ?db_type
WHERE {
?db_uri "http://www.w3.org/2002/07/owl\#sameAs" ?wd_uri.
filter contains(str(?wd_uri),"http://www.wikidata.org/entity/").
?d_uri a ?db_type.
filter contains(str(?db_type),"http://dbpedia.org/ontology/")
}
```

Questa query restituisce l'identificativo wikidata ed il corrispondente tipo/i; in particolare per ogni URI di Wikidata data in input, ricerca la corrispondente entry in DBpedia, controlla che il formato della URI di Wikidata sia corretta, ed infine restituisce il tipo controllando che anche questo valore sia nel formato corretto. Data la grande mole di entità nel nostro dataset iniziale, abbiamo eseguito queste query un po' alla volta, evitando di ricevere risposte negative dall'endpoint SPARQL di DBpedia. Le tipologia di classi presenti attualmente sono raffigurate all'indirizzo `http://mappings.dbpedia.org/server/ontology/classes/`.
Per quando riguarda le altre due informazioni extra, bisogna puntualizzare cosa si intende per entità principale in questo contesto. L'estrattore ha come obiettivo quello di estrarre il maggior numero di entità all'interno di videolezioni in un corso specifico, perciò le entità dovranno essere affini al contenuto del corso. Per questo motivo, partendo dalla corrispondente entità in Wikidata[13] e Wikipedia[14], si è percorso ogni link presente sulla pagina (ogni link presente è diretto ad un'altra entità che ha una relazione con quella di partenza) in modo ricorsivo, generando così un grafo a più livelli, dove ogni livello comprende un set di entità. Ovviamente all'aumentare del numero di livelli (profondità del grafo), il numero di entità aumenta velocemente. Questa ricerca è stata effettuata per i primi 4 livelli, calcolando un valore di similarità, tra 0 e 1, per ogni entità trovata, proporzionale al livello corrispondente (più si trova in alto nel grafo e vicino all'entità principale, più il valore si avvicinerà a 1). Cercando per così tanti livelli non si ha la certezza di prendere in considerazione solo entità effettivamente coerenti con l'entità principale, ma l'importante è trovare la maggior parte delle entità che effettivamente hanno un senso all'interno del corso; l'unica alternativa, sarebbe scrivere a mano i concetti principali e collegarli a Wikidata, ma oltre ad essere un lavoro lungo e poco scalabile, non si avrebbe mai la certezza di aver considerato tutte le entità di interesse.

[13]Per il corso 'Basi di dati' l'entità corrispondente sarà `https://www.wikidata.org/wiki/Q8513`, corrispondente a *database*

[14]Per il corso 'Basi di dati' l'entità corrispondente sarà `https://it.wikipedia.org/wiki/Base_di_dati`,corrispondente a *Base di dati*

5.4 Preparazione dei dati

Ricapitolando, da un lato si sono preparati i valori di Wikidata in locale, per essere facilmente raggiungibili, e dall'altra parte si sono creati dei corpus contenenti le informazioni di un corso registrato (transcript e slide). Come si può vedere dallo schema in fig.5.1, per poter confrontare i dati con quelli presenti su Wikidata, è necessaria una fase di *tokenizzazione*, operazione eseguita anche in [1] nella preparazione dei dati, ed in molti estrattori presenti in letteratura. Per tokenizzazione in questo ambito, si intende la divisione del corpus di input in singoli token, e succesivamente la creazione di stringhe, composte da questi token, in numero variabile; in particolare sono state create tutte le stringhe ordinate, da 1 a 10 token, considerando i segni di punteggiatura come veri e propri token. Ad esempio per la stringa:

La lezione di oggi sarà focalizzata sul linguaggio SQL.

la divisione in token restituisce il seguente risultato:

('la', 0, 1), ('lezione', 3, 9), ('di', 11, 12), ('oggi', 14, 17), ('sarà', 19, 22), ('focalizzata', 24, 34),('sul', 36, 38), ('linguaggio', 40, 49), ('sql', 51, 53), ('.', 54, 54)

Come si può vedere dall'esempio, il segno di punteggiatura è considerato a parte, mentre gli spazi tra le parole non vengono considerati. Inoltre è utile salvare insieme ai token, anche i relativi indici di inizio e fine. Infine, il risultato con la creazione delle stringhe è riportato in fig.5.7. Come si può vedere da questo esempio, sono

```
['lezione', 'oggi', 'focalizzata', 'linguaggio', 'sql', 'la lezione', 'lezione di', 'di oggi', 'oggi sarà', 'sarà f
ocalizzata', 'focalizzata sul', 'sul linguaggio', 'linguaggio sql', 'sql.', 'la lezione di', 'lezione di oggi', 'di
oggi sarà', 'oggi sarà focalizzata', 'sarà focalizzata sul', 'focalizzata sul linguaggio', 'sul linguaggio sql', 'l
inguaggio sql.', 'la lezione di oggi', 'lezione di oggi sarà', 'di oggi sarà focalizzata', 'oggi sarà focalizzata s
ul', 'sarà focalizzata sul linguaggio', 'focalizzata sul linguaggio sql', 'sul linguaggio sql.', 'la lezione di ogg
i sarà', 'lezione di oggi sarà focalizzata', 'di oggi sarà focalizzata sul', 'oggi sarà focalizzata sul linguaggi
o', 'sarà focalizzata sul linguaggio sql', 'focalizzata sul linguaggio sql.', 'la lezione di oggi sarà focalizzat
a', 'lezione di oggi sarà focalizzata sul', 'di oggi sarà focalizzata sul linguaggio', 'oggi sarà focalizzata sul l
inguaggio sql', 'sarà focalizzata sul linguaggio sql.', 'la lezione di oggi sarà focalizzata sul', 'lezione di oggi
sarà focalizzata sul linguaggio', 'di oggi sarà focalizzata sul linguaggio sql', 'oggi sarà focalizzata sul linguag
gio sql.', 'la lezione di oggi sarà focalizzata sul linguaggio', 'lezione di oggi sarà focalizzata sul linguaggio s
ql', 'di oggi sarà focalizzata sul linguaggio sql.', 'la lezione di oggi sarà focalizzata sul linguaggio sql', 'lez
ione di oggi sarà focalizzata sul linguaggio sql.', 'la lezione di oggi sarà focalizzata sul linguaggio sql.']
```

Figura 5.7: Risultato di esempio per la creazione delle stringhe di riferimento

stati eliminati tutti i token corrispondenti a *stopword*, cioè le parole più comuni usate nella lingua corrente. Queste stringhe di riferimento generate verranno confrontate con le caratteristiche delle entità presenti in Wikidata, cioè con tutte le label di tutte le entità descritte nel capitolo 5.3.2.

5.5 Ricercà di entità

Questa sezione riguarderà la parte centrale del progetto, corrispondente alla fase in cui, dati i token di input, e le labels di Wikidata, verrano cercate le coppie più simili tra loro. In pratica, ogni token andrà confrontato con ogni label, e se queste due saranno sufficiente *simili* (successivamente verrà chiarito meglio questo termine), allora verranno salvate per passare alla fase successiva. Data la grande mole di etichette (più di 40 milioni di stringhe) e di token per videolezioni (in media una lezione da un'ora e mezza genera circa cento mila token), effettuare tutti questi confronti ha richiesto un metodo efficiente, che restituisca risultati in un tempo ragionevole. A questo fine si sono provati tre metodi: una rete neurale in grado di trasformare una

stringa in un valore simile alla distanza di Levenshtein, una ricerca Nearest Neighbors utilizzando librerie Python, ed infine una clusterizzazione sui dati. Per molte di queste fasi abbiamo fatto uso del Cluster fornito dal Politecnico di Torino[15], le cui funzionalità si sono rivelate molto utili. Riguardo al primo metodo, si è provato a riprodurre l'architettura della rete neurale presentata in [21]. In quest'ultimo viene presentato un modello LSDE, un nuovo approccio per rappresentare stringhe che impara uno spazio in cui le distanze tra i punti proiettati sono correlate con la distanza di Levenshtein tra le stringhe originali. Questo metodo è stato scartato perchè l'analisi di questo sistema avrebbe impegato troppo tempo, ma non è da escludere che in futuro venga ripreso ed inserito all'interno della pipeline presentata.

5.5.1 Ricerca Nearest Neighbors

Dato che per confrontare stringhe non è possibile usare algoritmi che utilizzano distanze euclidee[16], si è cercata una rappresentazione idonea con cui trasformare stringhe in punti nello spazio. Per fare questo, abbiamo analizzato i caratteri presenti nelle etichette di Wikidata, ordinandole per numero di occorrenze. Il risultato sono state tutte le lettere dell'alfabeto, maiuscole e minuscole, i numeri, ed i segni di punteggiatura più comuni (questo controllo è stato necessario dato che nei valori di partenza di Wikidata esistono anche stringhe formate da simboli e caratteri non comuni). Di conseguenza, si è utilizzato un vettore di 37 celle, in cui le prime 25 celle (da 0 a 25) corrispondevano alle lettere dell'alfabeto (in maiuscolo), le successive 9 rappresentavano i numeri da zero a nove. Il valore di queste celle corrisponderà al conteggio della corrispettiva lettera all'interno dell'input, e se il carattere non sarà presente (segni di punteggiatura, spazi ecc..), allora verrà incrementata l'ultima cella del vettore. La fig 5.8 riporta un esempio di questa rappresentazione per la stringa *corso di basi dati*. Ultima operazione prima di passare alla ricerca vera e propria, è

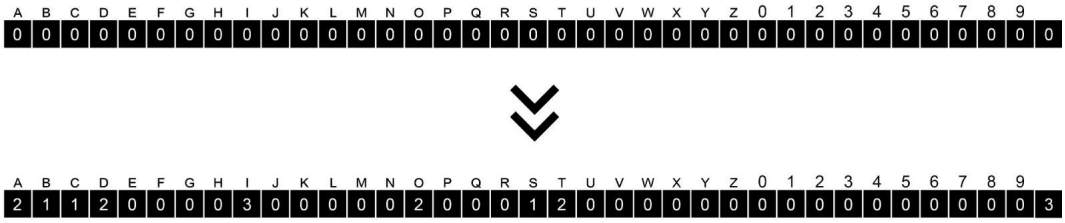


Figura 5.8: Esempio rappresentazione della stringa *corso di basi dati*

stata quella di ridurre la dimensionalità del vettore in modo da velocizzare l'intero processo. Per fare ciò è stato eseguita una PCA[17], in modo tale che il *variance ratio totale* (rappresenta quante informazioni sono attribuite a ciascuna delle componenti principali) sia almeno del 90%. In questo modo è stato possibile diminuire il numero di dimensioni da 36 a 8. A questo punto, utilizzando il cluster per trasformare ogni label di Wikidata nella corrispondente rappresentazione vettoriale spiegata in precedenza, e dopo aver applicato anche la PCA a quest'ultime, si è utilizzata una libreria

[15] `https://bigdata.polito.it/content/bigdata-cluster`

[16] La distanza euclidea è la distanza tra due punti, ossia la misura del segmento avente per estremi i due punti

[17] L'analisi delle componenti principali (in inglese principal component analysis o abbreviata PCA) è una tecnica per la semplificazione dei dati

Python che fornisce dei metodi di apprendimento basati sui vicini. Il principio alla base consiste nel trovare un numero predefinito di campioni di allenamento più vicini alla distanza dal nuovo punto e prevedere l'etichetta da questi. Il numero di campioni può essere una costante definita dall'utente, o variare in base alla densità locale dei punti. La distanza può, in generale, essere qualsiasi metrica: la distanza euclidea standard è la scelta più comune. Questi metodi di *Nearest Neighbors* sono noti come metodi di apprendimento automatico non generalizzati , poiché semplicemente "ricordano" tutti i dati di addestramento (eventualmente trasformati in una struttura di indicizzazione rapida come *Ball Tree* o *KD Tree*). Utilizzando questa funzionalità appena illustrata, è stato possibile creare questa struttura dati capace di, data una stringa di riferimento come input in rappresentazione vettoriale e ridotto con la PCA, di restituire gli n vicini più vicini, dove n è stato scelto uguale a 100. Infine per ognuno di questi vicini, viene applicata la distanza di Levenshtein[18] tra la rappresentazione originale del vicino, con l'input iniziale. Purtroppo questo metodo si è dimostrato inaffidabile, in quanto i risultato erano corretti, ma il tempo di elaborazione non era costante e variava in base alla lunghezza della stringa cercata, in particolare peggiorava nettamente per stringhe brevi.

5.5.2 Clusterizzazione dei dati

Quest'ultimo metodo, quello che è stato effettivamente usato nell'applicazione, consiste nel clusterizzare[19] tutte le etichette provenienti da Wikidata, in modo tale che ogni cluster sia il più omogeneo possibile rispetto al numero totale di elementi. In realtà, in generale il clustering viene effettuato direttamente sul set di dati effettivo, in questo caso invece, l'obiettivo è quello di creare una struttura efficiente per la ricerca dei vicini. Infatti una volta clusterizzate le labels di Wikidata, per confrontare una nuova stringa è sufficiente trovare il medoide più simile, ed infine all'interno del cluster trovato, effetuare una ricerca lineare con costo *O(n)*. Per poter seguire questa via, si è provato ad eseguire un K-Means[20] sulla rappresentazione vettoriale delle label, ma data la grande mole di dati, questo approccio probabilmente non sarebbe mai arrivato a convergere. Perciò, si è optato per scartare l'utilizzo di algoritmi che usavano centroidi e distanze euclidee come distanza di similarità (perciò non utilizzeremo più la rappresentazione vettoriale). La soluzione è stata creare un algoritmo di clustering incrementale, mostrato in 1, in cui i dati di input corrispondo alle etichette di Wikidata. Verrà utilizzata più volte l'equazione 5.3 che combina più volte la distanza di Levenshtein, dove a e b sono due stringhe, ed A e B sono le corrispondenti stringhe in maiuscolo:

$$distance1 = 0.2 * Levenshtein(a, b) + 0.8 * Sort\_Levenshtein(a, b) \tag{5.1}$$

$$distance2 = 0.2 * Levenshtein(A, B) + 0.8 * Sort\_Levenshtein(A, B) \tag{5.2}$$

[18]La distanza di Levenshtein è una misura per la differenza fra due stringhe, cioè il numero minimo di modifiche elementari che consentono di trasformare una nell'altra. Per modifica elementare si intende: la cancellazione di un carattere, la sostituzione di un carattere con un altro, o l'inserimento di un carattere.

[19]il clustering, o analisi dei gruppi, è un insieme di tecniche di analisi dei dati volte alla selezione e raggruppamento di elementi omogenei in un insieme di dati

[20]L'algoritmo K-means è un algoritmo di clustering partizionale che permette di suddividere un insieme di oggetti in K gruppi sulla base dei loro attributi

$$distance = 0.3 * distance1 + 0.7 * distance2 \tag{5.3}$$

Sort_Levenshtein, presnte in 5.1 e 5.2, applica Levenshtein dopo aver riordnato le parole all'interno della stringa in ordine alfabetico.

Le funzioni presenti sono:

- **computeClusters**: riceve in input *points*, una lista di n stringhe, e il numero di medoidi k desiderato. Essa richiama *computePairwiseDistances*, che è una funzione paralellizata sul Cluster, la quale restituisce, a partire da una lista di stringhe, una matrice quadrata di lato n, corrispondente all'applicazione dell'equazione 5.3 tra ogni coppia di stringhe. Successivamente tramite la funzione *kMedoids*[21], vengono calcolati i k medoidi[22] migliori utilizzando una libreria di clusterizzazione di Python. Da notare che una volta calcolata la matrice, il calcolo dei medoidi è un operazione velocissima, e soprattutto è indipendente dal valore di k. Infine restituisce i k medoidi calcolati.

- **reassignPoints**: essenzialmente richiama la funzione *computeNearestMedoid*, la quale è parallelizata sul cluster, e per ogni stringa in points, cerca la stringa più simile nella lista medoids, utilizzando 5.3, creando così la clusterizzazione vera e propria.

- **clustering**: è la funzione principale, e riceve come input le labels. La funzione ha tre ramificazioni, una volta entrata nel ciclo (prima di queste tre scelte effettua un ordinamento dei cluster per numero di elementi al proprio interno, alla riga 21). La prima di queste, righe 22-28, controlla se il numero di cluster è superiore al massimo consentito (nel nostro caso 10000); se lo è, elimina i cluster con meno elementi e riassegna i punti appartenenti a quei cluster ai restanti medoidi. La seconda ramificazione, righe 30-33, controlla se il cluster col maggior numero di punti, ha un numero di elementi minore della soglia massima (100000); se si, allora l'algoritmo è terminato con successa e ritorna la struttura con la clusterizzazione completa. Infine l'ulimo ramo, da 35 a 43, a cui si accede solo se il numero di cluster è minore della sogna, e la condizione di terminazione (riga 31) non è soddisfatta. Per ogni cluster il cui numero di elementi è superiore alla soglia (50000), vengono selezionati max point(dove max è 10000), vengono riordinati in modo casuale, viene compiuta la funzione *computeClusters*, ed i medoidi risultanti andranno a sostituire quello precedente. Infine, avendo l'elenco di tutti i nuovi medoidi, insieme a quelli vecchi validi, viene richiamata *reassignPoints*.

E' stato necessario adottare un approccio di questo tipo in cui, ad ogni ciclo di clusterizzazione, ogni stringa del set iniziale veniva riassegnata, in quanto non vale la proprietà transitiva, cioè quando un medoide viene sostituito da altri 100 generati dai punti al suo interno, può capitare che punti asseganti esternamente, dopo questa fase, risultino più vicino ad un uno dei nuovi medoidi.

[21] K-medoids è un algoritmo di clustering partizionale correlato all'algoritmo K-means.

[22] I medoidi sono oggetti rappresentativi di un set di dati o un cluster con un set di dati la cui dissomiglianza media rispetto a tutti gli oggetti nel cluster è minima.I medoidi sono simili nel concetto ai centroidi , ma essi sono sempre limitati a essere membri del set di dati.

Algorithm 1 clustering medoids

```
input : allLabels
output : clusters
procedure computeClusters(points,k)
    D ← computePairwiseDistances of points
    medoidsIndex ← kMedoids of D,k
    return list of points matching with medoidsIndex
procedure reassignPoints(points,clusters,medoids=False)
    if not medoids then
        medoids ← list of key in clusters
    clustering ← computeNearestMedoid of points,medoids
    return clustering
procedure clustering(allLabels)
    maxMedoids ← 10000
    maxValuePerCluster ← 100000
    max ← 10000
    k ← 100
    thresholdCluster ← 50000
    strings ← allLabels
    clusters ← allLabels
    while True do:
        sortedClusters ← sortClusterByLenght of clusters
        if len(clusters) > maxMedoids then
            points ← list()
            while i < len(clusters)-maxMedoids do
                medoid, numberElements ← sortedClusters[i]
                points ← add points in clusters[medoid]
                delete clusters[medoid]
                i ← i + 1
            clusters ← reassignPoints of points,clusters
        else
            medoid, numberElements ← sortedClusters[−1]
            if numberElements < maxValuePerCluster then
                save clusters
                exit
            else
                medoids ← set()
                for all medoid,numberElements in sortedClusters do
                    if numberElements > thresholdCluster then
                        points ← points in clusters[medoid]
                        delete clusters[medoid]
                        shuffle of points
                        medoid ← computeClusters of points[:max],k
                        medoids ← add medoid
                clusters ← reassignPoints of strings,clusters,medoids
```

5.5.3 Estrazione entità

Una volta ottenuta la clusterizzazione, è stato possibile ricercare all'interno di essa l'etichetta più simile corrispondente ad ogni stringa di riferimento, in modo efficiente. Per fare ciò è stato utilizzato nuovamente il Cluster del Politecnico. Perciò per ogni stringa da confrontare, viene fatta una ricerca lineare sul set di medoidi finali, trovando il corrispondente medoide più *simile*, cioè quello corrispondente al valore minore dato dall'equazione in 5.3 e generato dei file corrispodenti. In seguito, per ogni stringa associata ad un medoide, sono state ricercate tutte le etichette (presenti nel medoide), che avessero una misura di similarità di almeno 0.7, la quale è stata calcolata usando una libreria Python che fornisce diverse funzioni per il calcolo di similarità tra stringhe, riportate in 5.6, dove a e b sono due generiche stringhe, mentre A e B sono le corrispondenti stringhe in maiuscolo .

$$sim1 = 0.8 * ratio(a, b) + 0.2 * token\_sort\_ratio(a, b) \tag{5.4}$$

$$sim2 = 0.8 * ratio(A, B) + 0.2 * token\_sort\_ratio(A, B) \tag{5.5}$$

$$sim = 0.3 * sim1 + 0.7 * sim2 \tag{5.6}$$

In 5.4 e 5.5 vengono utilizzate due funzioni, *ratio* e *token_sort_ratio*: la prima calcola la similarità tra due stringhe senza maniporlarle, la seconda invece ordina ogni token all'interno delle due stringhe prima di calcolarne la similarità. Entrambe sfruttano al loro interno la distanza di Levenhstain e restituiscono un valore tra 0 e 1. Successivamente, avendo trovato per ogni stringa di riferimento le etichette più simili (molte di queste, non avendo nessuna label più simile di 0.7, sono stati scartati), sono state cercate in locale le entità corrispondenti a tali label, accedendo al database MongoDB dove si erano precedentemente salvate tutte le informazioni. Infine, per ogni set di label candidate per una stringa di riferimento, si mantengono tutte le entità corrspondenti a label in cui la misura di similarità è uguale a 1, ed in aggiunta a queste, anche le prime 10 entità con i valori corrispondenti di similarità piu alti.

5.6 Filtraggio entità

La parte finale di questa pipeline, consiste nel selezionare, tra le possibili entità candidate associate ad una menzione, l'entità migliore in base al contesto di utilizzo. Per effettuare questa *scelta* si utilizzzeranno due alberi di decisione. Questi *decision trees* (DT) sono metodi di apprendimento supervisionato utilizzato per la classificazione e la regressione. L'obiettivo è quello di creare un modello che preveda il valore di una variabile target apprendendo semplici regole decisionali dedotte dalle caratteristiche dei dati. Questi alberi decisionali presentano numerose caratteristiche positive. Sono semplici da capire ed interpretare, inoltre l'albero risultante può essere visualizzato. Richiede una scarsa preparazione dei dati, dato che altre tecniche richiedono solitamente la normalizzazione dei dati. Il costo dell'utilizzo dell'albero (ovvero la previsione dei dati) è logaritmico nel numero di punti dati utilizzati per addestrare l'albero. In generale però, c'è il rischio di generare alberi troppo complessi, che non generalizzano bene i dati. Questa situazione è detta overfitting[23].

[23]Si parla di overfitting quando un modello statistico molto complesso si adatta ai dati osservati (il campione) perché ha un numero eccessivo di parametri rispetto al numero di osservazioni.

La prima fase ragiona dividendo le stringhe di riferimento da disambiguare in singoli token. Si consideri la stringa di riferimento s, a cui sono associate 2 entità candidate, e_1 ed e_2. Supponendo che s sia composta da 3 token diversi, si estrapolano quest'ultimi, e si valuta una probabilità per ognuno di essi, basata su quanto il token è corretto all'interno di quell'entità specifica. In questo caso, avendo 3 token t_1, t_2 e t_3, viene calcolata una probabilità per ognuno di essi, una per ogni entità candidata, e come step finale l'albero di decisione restituisce un valore che corrisponde ad una media delle probabilità dei singoli token. In questa fase viene considerata una caratteristica aggiuntiva per ogni token molto utile, cioè i *POS tagger*. Questi tag di parte del discorso, chiamato anche tagging grammaticale o disambiguazione di categorie di parole, è il processo di marcatura di una parola in un testo (corpus) come corrispondente a una parte specifica di parola, basato sia sulla sua definizione che sul suo contesto, cioè sulla sua relazione con parole adiacenti e correlate in una frase o paragrafo. Con questo sistema si può associare ad ogni token un determinato tipo associato alla sua posizione nel testo, che sarà utile ai fini della disambiguazione.
La seconda fase, effettua la disambiguazione finale, restituendo per ogni stringa di riferimento, al massimo una sola risposta positiva (non è possibile che l'albero restituisca due risultati postitivi per la stessa stringa, ma è possibile che *decida* che nessuno tra le entità candidate sia quella giusta). Per effettuare la *disambiguazione* utilizza le seguenti *features*:

- il valore di similarità dell'entità proveniente dal grafo di Wikipedia;
- il valore di similarità dell'entità proveniente dal grafo di Wikidata;
- la similarità tra la stringa di riferimento e la label dell'entità candidata trovata durante l'estrazione dell'entità;
- la ligua della label dell'entità candidata trovata durante l'estrazione dell'entità;
- la descrizione dell'entità
- la tipologia dell'entità, convertendo i tipi con una codifica *one-hot-encoding*[24] (necessaria dato che ad una entità possono essere associati più tipi);
- la media delle similarità tra le label inglesi dell'entità e la stringa candidata;
- la media delle similarità tra le label italiane dell'entità e la stringa candidata;
- la media delle similarità tra le label alternative inglesi dell'entità e la stringa candidata;
- la media delle similarità tra le label alternative italiane dell'entità e la stringa candidata;
- la media delle similarità tra le label preferite inglesi dell'entità e la stringa candidata;
- la media delle similarità tra le label preferite italiane dell'entità e la stringa candidata;

[24]Una codifica *one-hot-encoding* è un processo mediante il quale le variabili categoriali vengono convertite in una forma che può essere fornita ad algoritmi ML per effettuare un lavoro di previsione.

- il numero di entità candidate per la stringa di riferimento in questione;
- il numero di entità uguali presenti nelle stringhe di riferimento corrispondenti a 20 caratteri prima e 20 caratteri dopo la stringa di riferimento. Questo parametro serve per valutare se in un range intorno alla stringa, è presente la stessa entità come entità candidata;
- il valore di probabilità calcolato nella fase precedente.

Verrano ora riportati i modelli risultanti di questa fase. Per la valutazione del lavoro si utilizzeranno 3 dataset differenti:

- il primo ottenuto dalle videolezioni del corso *Basi di dati*, estrapolando casualmente 10 frammenti per ogni lezione da 30 parole ciascuno, per un totale di 430 frammenti. Di questi, 300 sono stati assegnati al *train set*, e i restanti 130 al *test set*. Generati i due set sono stati successivamente annotati come spiegato in 6.1.1
- il secondo è ottenuto dal precedente dataset, tenendo in considerazione solo le entità inerenti effettivamente al corso.
- il terzo comprende 1300 tweet provenienti da Twitter, divisi in 1000 tweet per il *train set* e *test set*. Questi dati sono stati ottenuti da una challenge proposta da EVALITA 2016, illustrata nel capitolo 7.1

Gli alberi risultanti sono mostrati in fig. 5.9, 5.10 e 5.11. Osservando gli alberi

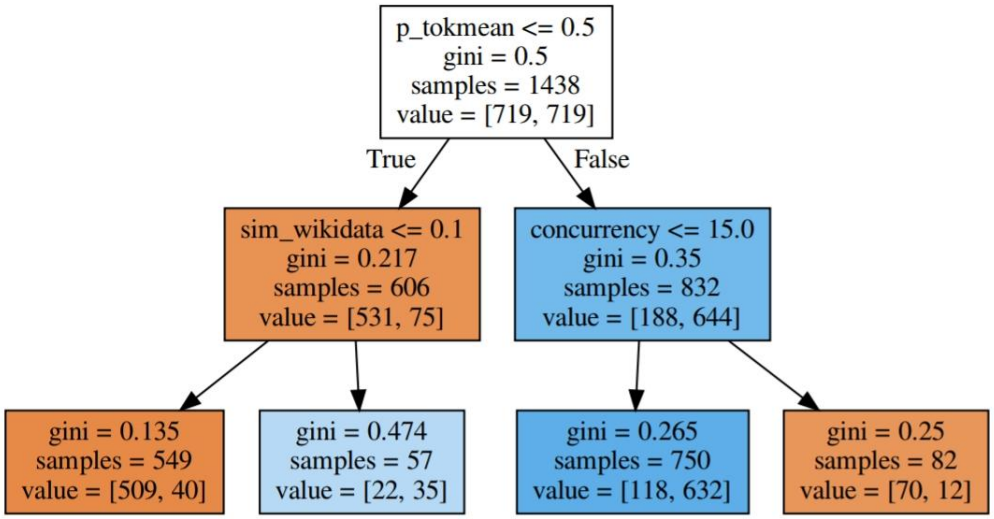


Figura 5.9: Albero di decisione per il dataset dei frammenti

di decisione, è facile notare come il primo valore effettivamente usato corrisponde a *p_tokmean*, cioè il valore proveniente dalla prima fase descritta in precdenza. Altro risultato importante è rappresentato dal valore *sim_wikidata*, corrispondente al valore di similarità del grafo di Wikidata presentato in 5.3.4. Questo valore compare solo nei due modelli corrsipondenti ai frammenti, in quanto solo in questo caso esiste un contesto specifico (Basi dati), mentre il contesto all'interno di Twitter è generico. Questo sarà il principale motivo per cui il modello presentato non sarà in grado di adattarsi bene al dataset dei tweet.

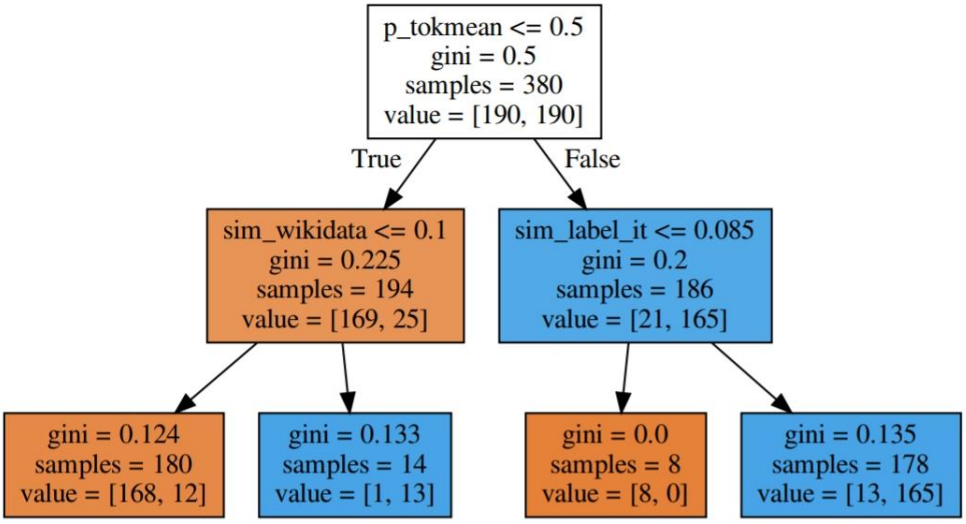


Figura 5.10: Albero di decisione per il dataset dei frammenti filtrati

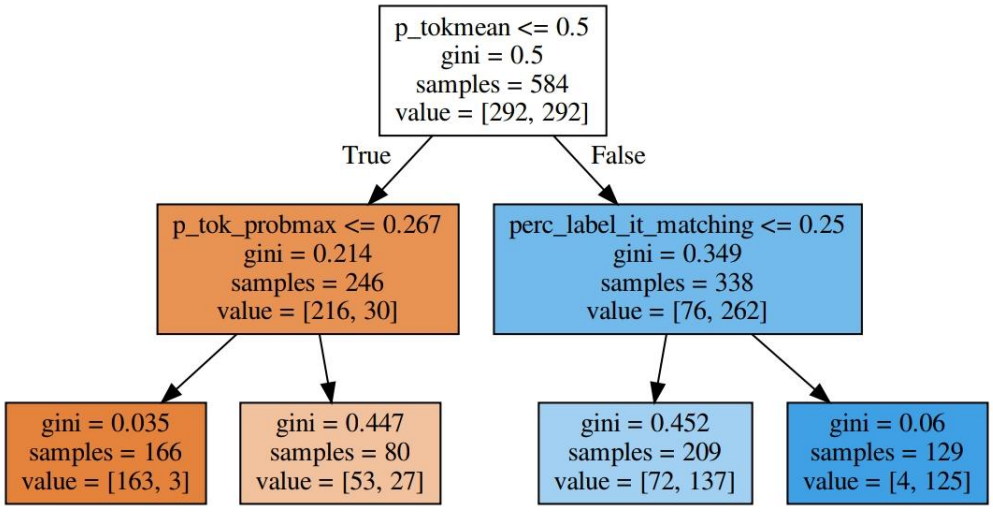


Figura 5.11: Albero di decisione per il dataset di Twitter

Capitolo 6

Ground truth

In questo capitolo si farà un accenno alle fasi e alle applicazioni utilizzate, per creare due **ground truth**[1] diverse per valutare il progetto realizzato. I risultati verranno presentati nel capitolo successivo. Questo progetto è nato dall'esigenza di creare un sistema che aiutasse gli studenti nella ricerca di contenuti didattici all'interno di un corso videoregistrato. Per questo motivo il primo dataset preso in considerazione è stato il corso *Basi di dati*, già citato più volte durante il report, in modo da effettuare una valutazione concreta. In aggiunta a quest'ultimo (che rimane l'obiettivo finale), si è testato questo approccio anche su un altro dataset, sempre in italiano, proveniente da Twitter[2].

6.1 Frammenti di videolezioni

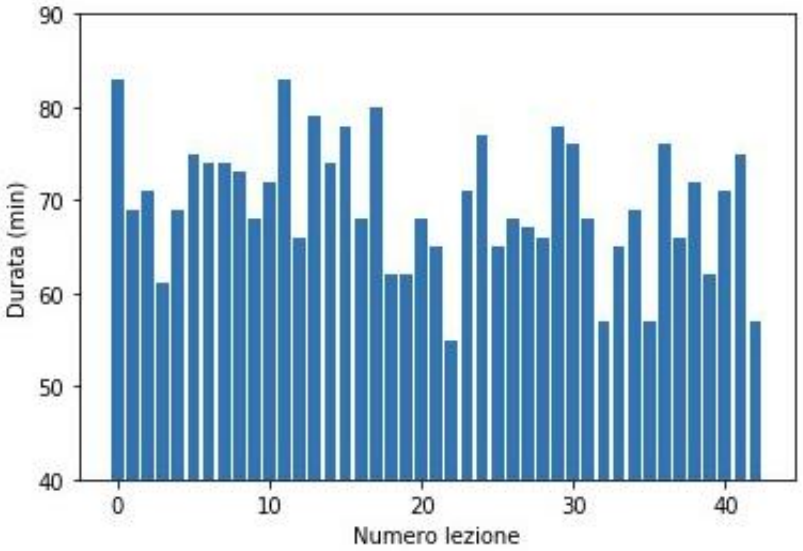


Figura 6.1: Durata videolezioni nel corso *Basi di dati*.

[1]La ground truth è un termine usato in vari campi per riferirsi alle informazioni fornite dall'osservazione diretta (cioè evidenza empirica) in contrapposizione alle informazioni fornite dall'inferenza.

[2]`https://twitter.com/`

Come primo set di validazione verranno utilizzate le videolezioni disponibili per il corso *Basi di dati*. La durata delle lezioni è riporta in fig. 6.1, in cui la durata media è di 70 minuti. Annotare a mano tutte le 43 videolezioni sarebbe un lavoro troppo lungo e dispendioso, soprattuto se fatto da una persola sola. Per questo motivo, sono state selezionate casualmente in ogni lezione, 10 frammenti composti ognuno da 30 parole, in modo tale da formare un totale di 430 frammenti. Come detto nel capitolo 5.6, sono stati divisi in *train* e *test* (rispettivamente 300 e 130).

6.1.1 Tool per l'annotazione

Per questa fase di annotazione si è utilizzato il tool gratuito di **Brat**[3]. *Brat* è uno strumento web per l'annotazione del testo, cioè per aggiungere note a documenti di testo esistenti[33]. In particolare, è progettato per l'annotazione strutturata, in cui le note non sono testo a mano libera ma hanno una forma fissa che può essere elaborata e interpretata automaticamente da un computer. In questo progetto, *brat* è utile per avere un interfaccia facile da utilizzare per annotare i documenti ed inoltre offre la possibilità di caricare localmente i file di interesse. La raccolta è una cartella che contiene il testo, diviso in file con estensione *.txt*, che devono essere annotati. Per ciascuno di questi file, è necessario aggiungere un file di annotazione con suffisso *.ann* nella cartella con lo stesso nome base; nella fase iniziale questo file sarà vuoto, e verrà succesivamente compilato dal tool a seguito delle annotazioni effettuate tramite l'interfaccia. Ad esempio, in una generica cartella, se fosse presente il file *example.txt*, sarebbe necessario trovare il suo corrispettivo file di annotazione *example.ann*. Le annotazioni create in *brat* sono memorizzate nei file di annotazione separatamente dal testo del documento, che non viene mai modificato dallo strumento. Tutte le annotazioni seguono la stessa struttura di base: ogni riga contiene un'annotazione e ogni annotazione riceve un ID che appare per primo sulla linea, separato dal resto dell'annotazione da un singolo carattere TAB. Il resto della struttura varia in base al tipo di annotazione. Tutti gli ID di annotazioni sono costituiti da un singolo carattere maiuscolo che identifica il tipo di annotazione e un numero. I caratteri ID iniziali si riferiscono ai tipi di annotazione come segue:

- T (*confini della annotazione*). Le annotazioni legate al testo sono una categoria importante di annotazioni correlate alle annotazioni di entità e di eventi. L'annotazione legata al testo identifica un intervallo specifico di testo e assegna a esso un tipo. La struttura è sempre la stessa: il primo elemento è il numero della annotazione (ad esempio T1), seguito dalla annotazione principale e infine dalla menzione corrispondnete nel testo, dicise dal carattere TAB. L'annotazione principale viene fornita come tripla separata da SPACE (tipo, inizio-offset, fine-offset). L'offset iniziale è l'indice del primo carattere dello span annotato nel testo (file *.txt*), cioè il numero di caratteri nel documento che lo precede. L'offset finale è l'indice del primo carattere dopo l'intervallo annotato. Pertanto, il carattere nella posizione di fine-offset non è incluso nello span annotato.

- R (*relazioni*). Le relazioni binarie hanno un ID univoco e sono definite dal loro tipo (ad es. Origin, Part-of) e dai loro argomenti.

[3] `http://brat.nlplab.org/index.html`

- E (*evento*). Ogni annotazione di evento ha un ID univoco ed è definita in base al tipo (ad esempio MERGE-ORG), all'evento trigger (il testo che indica l'evento) e agli argomenti.
- # (*note*). Le note forniscono un modo per associare il testo a mano libera al documento o ad un'annotazione specifica. Le righe delle note iniziano con il simbolo #.

In questo progetto, verranno utilizzate solo le prime e ultime. Infatti si è interessati ad annotare delle parti di testo, senza associare il corrispondente tipo dato che per ora non ne terremo conto, con le rispettive note contenenti la corrispondente entità su Wikidata. In fig. 6.2 viene riportata un esempio di frase annotata. Come si

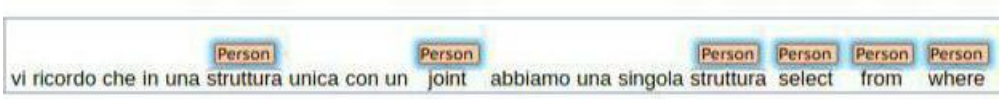


Figura 6.2: Esempio di frase annotata.

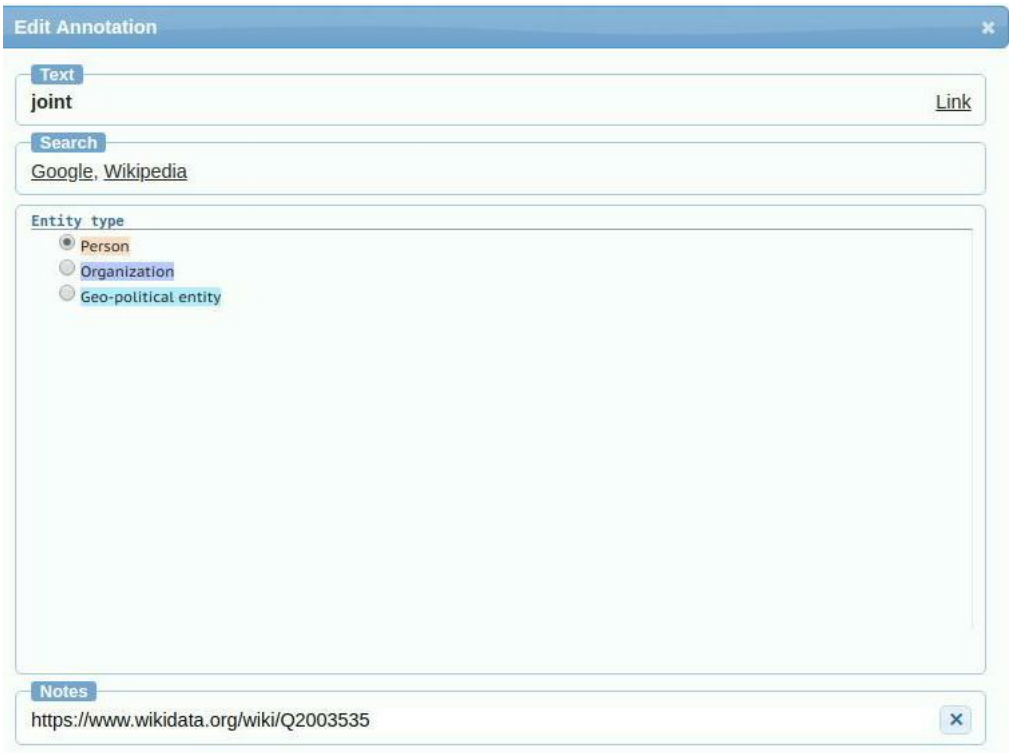


Figura 6.3: Esempio di menù per l'annotazione.

può vedere dalla figura, ogni entità annotata ha come riferimento il tipo *Person*, ma in reltà questa è solo l'impostazione di default in quanto come detto in precedenza, per ora non siamo interessati alla tipologia dell'entità. In fig. 6.3 viene riportato un tipico menù che compare dopo aver selezionato una determinata menzione. In questa finestra è possibile selezione il tipo (di default viene selezionato *Person*, ed inoltre agendo su un file di configurazione è possibile aggiungere tipologie a piacimento), ed inserire delle note; in questo caso infatti la menzione *joint*, che fa chiaramente riferimento all'operatore SQL *JOIN*, perciò è stata annotata con la rispettiva entità di Wikidata disponibile all'indirizzo `https://www.wikidata.org/wiki/Q2003535`. Per

questa frase presa in considerazione, il corrispondente file annotato sarà strutturato come segue:

```
T1 Person 22 31 struttura
#1 AnnotatorNotes T1 https://www.wikidata.org/wiki/Q6671777
T2 Person 71 80 struttura
#2 AnnotatorNotes T2 https://www.wikidata.org/wiki/Q6671777
T3 Person 81 87 select
#3 AnnotatorNotes T3 https://www.wikidata.org/wiki/Q1164001
T4 Person 88 92 from
#4 AnnotatorNotes T4 https://www.wikidata.org/wiki/Q3304069
T5 Person 93 98 where
#5 AnnotatorNotes T5 https://www.wikidata.org/wiki/Q2668363
T6 Person 142 149 tabelle
#6 AnnotatorNotes T6 https://www.wikidata.org/wiki/Q496946
T7 Person 45 50 joint
#7 AnnotatorNotes T7 https://www.wikidata.org/wiki/Q2003535
```

Perciò un passo fondamentale una volta identificata la menzione da annotare, è la ricerca di tutte le possibile candidate, ed infine collegare la menzione all'entità Wikidata più consona in base al contesto e al senso della frase, assegnando l'identificativo correlato. Per eseguire la ricerca dell'entità, si è utilizzato il motore di ricerca di Wikidata, con la menzione da ricercare come input. Tuttavia, quando la menzione non corrisponde esattamente all'etichetta dell'entità, il motore di ricerca di Wikidata non è in grado di trovare l'entità di disambiguazione appropriata. Per questo motivo, per risolvere questo problema si può utilizzare una ricerca su Google, che di solito restituisce la corretta pagina di Wikipedia con l'etichetta corretta. Usando questa etichetta come input del motore di ricerca di Wikidata è possibile recuperare l'entità corretta e l'URI associato. Sovente anche utilizzare la traduzione inglese della menzione cercata può essere d'aiuto.

6.2 NIF files

Per poter caricare i dataset su GERBIL(4.1), è stato necessario convertrli in formato NIF, cioè *Natural Language Processing Interchange Format*. NIF è un ontologia che descrive le stringhe. Per illustrarne la struttura verrà utilizzata la seguente frase campione, presente del dataset di Twitter:

Totti, Roma unito e rema da stessa parte: Capitano: cerchiamo di arrivare insieme uniti fino alla fine

NIF è un modo per indirizzare stringhe arbitrarie con URI. Le stringhe da descrivere sono generalmente identificate tramite offset che iniziano con 0, contando gli spazi tra i caratteri. Ogni documento è espresso come risorsa **nif:Context**. Quindi è possibile indirizzare il documento in questo modo:

```
<http://example.org/document0#char=0,102>
a nif:String , nif:Context , nif:RFC5147String ;
nif:isString """Totti, Roma unito e rema da stessa parte: Capitano:
cerchiamo di arrivare insieme uniti fino alla fine."""^^xsd:string ;
```

```
nif:beginIndex "0"^^xsd:nonNegativeInteger ;
nif:endIndex "102"^^xsd:nonNegativeInteger .
```

Il testo del documento è contenuto nella proprietà **nif:isString** obbligatoria. **nif: beginIndex** e **nif: endIndex** sono ulteriormente necessari per indicare gli offset della stringa. Le sottostringhe possono ora essere annotate dai rispettivi offset di stringa in riferimento al **nif:Context**, costruendo una struttura per ogni annotazione. La stringa della frase è contenuta nella proprietà **nif:anchorOf** obbligatoria. La proprietà **nif:referenceContext** collega la risorsa al documento.

```
<http://example.org/document0#char=0,5>
a nif:String , nif:Word , nif:RFC5147String ;
nif:anchorOf """Totti"""^^xsd:string ;
nif:beginIndex "0"^^xsd:nonNegativeInteger ;
nif:endIndex "5"^^xsd:nonNegativeInteger ;
nif:referenceContext <http://example.org/document0#char=0,102> ;
itsrdf:taIdentRef <http://dbpedia.org/resource/Francesco_Totti> .

<http://example.org/document0#char=42,50>
a nif:String , nif:Word , nif:RFC5147String ;
nif:anchorOf """Capitano"""^^xsd:string ;
nif:beginIndex "42"^^xsd:nonNegativeInteger ;
nif:endIndex "50"^^xsd:nonNegativeInteger ;
nif:referenceContext <http://example.org/document0#char=0,102> ;
itsrdf:taIdentRef <http://dbpedia.org/resource/Francesco_Totti> .

<http://example.org/document0#char=7,11>
a nif:String , nif:Word , nif:RFC5147String ;
nif:anchorOf """Roma"""^^xsd:string ;
nif:beginIndex "7"^^xsd:nonNegativeInteger ;
nif:endIndex "11"^^xsd:nonNegativeInteger ;
nif:referenceContext <http://example.org/document0#char=0,102> ;
itsrdf:taIdentRef <http://dbpedia.org/resource/A.S._Roma> .
```

In queste strutture contenenti le annotazioni vere e proprie delle menzioni, è presente anche **itsrdf:taIdentRef**, usato per collegarla a una risorsa di entità esterna, come una risorsa DBpedia.

Capitolo 7

Valutazione del Lavoro

In questo capitolo si analizzeranno i risultati ottenuti, sia per quanto riguarda i tweet di Twitter, sia per i frammenti annotati. È importante sottolineare il seguente concetto: il punto di partenza è stato cercare e realizzare un modello che si adattasse bene in condizioni sfavorevoli come quelle presentate dai transcript, perciò l'obiettivo principale era quello di riuscire ad ottenere risultati migliori rispetto agli estrattori presenti in letteratura presentati al capitolo 4. Con la valutazione del modello, applicato al dataset di Twitter, si è voluta testare una condizione diversa da quella di partenza, probabilmente sapendo di ottenere risultati non positivi, ma potendo così analizzare pregi e difetti del sistema creato, e poter progettare così delle migliorie mirate per il futuro.

7.1 Design degli esperimenti

Per la valutazione del progetto si sono utilizzati due dataset diversi: il primo proveniente dalle lezioni annotate di *Basi di dati*, ed il secondo proveniente da Twitter.
Per quanto riguarda le lezioni, si sono scelte randomicamente 10 frasi composte ognuna da 30 parole, per ognuna delle 43 lezioni disponibili. Per questa fase è stato utilizzato il tool di annotazione presentato in 6.1.1. Numericamente sono state annotate:

- 353 entità differenti;
- 2148 entità totali;
- una media di 50 entità a lezione e 5 entità a frammento.

Per quanto riguarda la creazione *train* e *test* set, si sono selezionati in ogni lezione 7 e 3 frammenti randomicamente, per un totale di 300 e 130 frammenti totali rispettivamente. In questa *ground truth*, sono presenti sia entità generali, che entità inerenti al corso.
Per questa ragione, si è creato un altro dataset, contenente solo entità effettivamente di interesse, in modo tale da valutare il progetto anche in questa situazione. Per la creazione del train e del test set, si sono mantenute le stesse identiche divisioni appena presentate. In aggiunta però, si è eseguito un filtraggio delle entità, analizzandole una ad una, ed eliminando dai due dataset quelle non inerenti al corso *Basi di dati* in base alla nostra esperienza. Il risultato di questa operazione ha mostrato

un numero totale di entità uguale a 84, rispetto alle 353 iniziali. Entrambi questi dataset sono stati annotati utilizzando Wikidata come base di conoscenza, ma per poter confrontare i risultati con GERBIL, il risultato delle predizioni effettuate dal modello presentato, è stato necessario convertire i dati con gli indirizzi corrispondenti su DBpedia[1], utilizzando una query SPARQL[2].
Per quanto riguarda il dataset di Twitter, i valori sono stati ottenuti da EVALITA. **EVALITA**[3] è una campagna di valutazione periodica di *Natural Language Processing* (NLP) e strumenti vocali per la lingua italiana. L'obiettivo generale di EVALITA è di promuovere lo sviluppo di tecnologie linguistiche e vocali per la lingua italiana, fornendo una struttura condivisa in cui diversi sistemi e approcci possono essere valutati in modo coerente. Nel 2016, ha proposto NEEL-IT[4], una *challenge* che ha il compito di effettuare delle valutazione nel contesto dei tweet italiani. In particolare, consiste nell'annotazione di ogni menzione di entità con nome (come persone, ubicazioni, organizzazioni e prodotti) in un testo collegandolo a una base di conoscenza (DBpedia). I dati forniti consistono in 1000 tweet per il *train*, e 300 per il *test* set. In particolare sono presenti:

- 458 entità differenti
- 751 entità totali

Il seguente esempio è preso dal *train* set:

1h30 di ricordi indimenticabili della stagione passata..quest'anno resterà nella storia!Quanto amo questa squadra #finoallafine #forzajuve

Questo set presenta però alcune caratteristiche differenti rispetto ai transcript:

- ogni entry corrisponde ad un generico tweet;
- questi twitter sono in genere strutturalmente ben scritti e con la presenza di punteggiatura;
- i frammenti contengono sovente trascrizioni non corrette, mentre in questi tweet la situazione peggiore è rappresentata da una abbreviazione usata nel linguaggio comune (ad esempio per citare Facebook[5], è possibile utilizzare *face* o *fb*);
- La mancanza totale di un contesto specifico.

In [32] viene descritto il progetto, in particolare agli obiettivi da raggiungere e alle metriche di valutazione utilizzate. Alla competizione hanno partecipato 6 team, e la descrizione del loro lavoro è riportata all'interno di [32]. In particolare, ai partecipanti era richiesto di:

- Riconoscere il tipo di ogni menzione di entità che appare nel testo di un tweet;

[1] `https://wiki.dbpedia.org/`
[2] `https://www.w3.org/TR/rdf-sparql-query/`
[3] `http://www.evalita.it/`
[4] `http://www.evalita.it/2016/tasks/neel-it`
[5] `https://www.facebook.com/`

- Disambiguare e collegare ogni menzione al DBpedia canonizzato 2015-10, che è usato come referente *Knowledge Base*;

- Raggruppare insieme le entità non collegabili, che sono taggati come NIL, al fine di fornire un identificatore unico per tutte le citazioni che fanno riferimento alla stessa entità nominata.

Per la valutazione verrà considerato solo il secondo punto. In aggiunta ai risultati complessivi della competizione, presentati in [32] insieme a degli accenni sui metodi da loro utilizzati, è stato pubblicato un articolo dal Politecnico di Bari [44], con il quale sarà possibile confrontare anche i risultati parziali.

7.2 Metriche di valutazione

Prima di passare alla vera e propria fase di valutazione, è necessaria breve panoramica sulle metriche utilizzate per valutare i modelli tra di loro. In questo contesto si hanno due classi a disposizione: l'*actual class*, che comprende i risultati corretti della fase di disambiguazione (cioè la ground truth spiegata nel capitolo 6), e la *predict class*, che comprende i risultati predetti dal modello in questione. Definiamo ora quattro combinazioni: TP (true positive, cioè veri positivi), FN (false negative, cioè falsi negativi), FP (false positive, cioè falsi positivi) e TN (true negative, cioè veri negativi). TP e TN sono le osservazioni che sono correttamente previste. Falsi positivi e falsi negativi si verificano quando la classe reale è in contraddizione con la classe predetta. La fig. 7.1 raffigura le possibili combinazioni. Più in dettaglio:

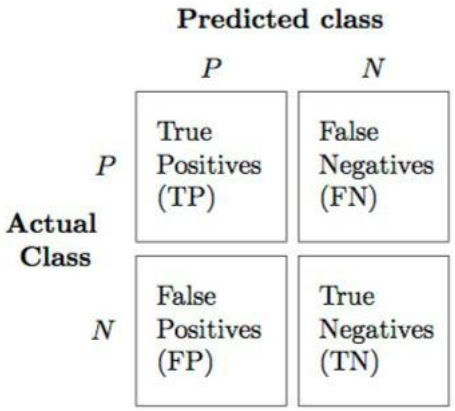


Figura 7.1: Schema delle possibili combinazioni dei risultati.

- True Positives (TP). Questi sono i valori positivi previsti correttamente, il che significa che il valore della classe effettiva è uguale al valore della classe predetta;

- True Negatives (TN). Questi sono i valori negativi previsti correttamente, il che significa che il valore della classe predetta è stato scartato in modo corretto;

- False Positives (FP). Quando la classe effettiva non è presente nel set finale e la classe predetta da come risultato l'opposto

- False Negative (FN). Quando la classe effettiva non è stata correttamente predetta.

Utilizzando questi 4 valori è possibile ora definire le tre metriche che si andranno ad utilizzare:

- **Recall**. Il richiamo è il rapporto tra le osservazioni positive correttamente previste e tutte le osservazioni nella classe effettiva positiva dell'*actual class*. L'equazione corrispondente è data da 7.1.

$$R = TP/(TP + TN) \tag{7.1}$$

- **Precisione**. La precisione è il rapporto tra le osservazioni positive correttamente previste e le osservazioni positive totali previste. L'equazione corrispondente è data da 7.2.

$$P = TP/(TP + FP) \tag{7.2}$$

- **F1 Score**. La F1 è la media ponderata di Precisione e Recall. Pertanto, questo punteggio prende in considerazione sia i falsi positivi che i falsi negativi. L'equazione corrispondente è data da 7.3.

$$P = 2 * (R * P)/(R + P) \tag{7.3}$$

7.3 Valutazione dei frammenti

In questo paragrafo si analizzeranno i risultati ottenuti utilizzando il *test set* ottenuto dai frammenti annotati, che sono 130. Dopo aver creato il corrispondente file NIF, si è caricato il dataset su GERBIL, e la tabella 7.1 mostra i risultati. Come si può ve-

Tabella 7.1: Risultati di GERBIL per il test set dei frammenti.

	F1 score	**Precisione**	**Recall**
AGDISTIS	0,043	0,043	0,043
AIDA	0,0342	0,11	0,02
Babelfy	0,08	0,20	0,05
DBpediaSpotlight	0,06	0,1	0,04
FOX	0	0	0
PBOH	0,17	0,17	0,17

dere dai risultati in 7.1, i valori in generale sono molto bassi. Questa è stata una delle motivazioni portanti nello sviluppo di un nuovo modello, che andasse bene con questo corpus, dato che gli estrattori già presenti in letteratura non erano sufficienti. Infatti, la mancanza di punteggiatura, la trascrizione non corretta di alcune parole e la mancanza di un contesto generale, ha condizionato negativamente i risultati. FOX, che in genere funziona discetamente, ha restituito solo valori nulli. I risultati ottenuti dal modello creato sono riportati nella tabella 7.6. È possibile visionare l'esperimento all'indirizzo `http://gerbil.aksw.org/gerbil/experiment?id=201811270001` Il risultato è stato positivo, in quanto in questa prima fase di costruzione del modello, si voleva arrivare ad avere alti valori di recall. In questo modo, partendo da

Tabella 7.2: Risultati del modello per il test set dei frammenti.

	F1 score	Precisione	Recall
Modello proposto	0,32	0,20	0,84

questo, sarà possibile migliorare in futuro la fase di disambiguation, in modo tale da alzare anche il valore di precisione, e di conseguenza di F1. Come detto in precedenza, in questo dataset di *test*, sono presenti anche entità al di fuori del contesto di Basi di dati. I risultati ottenuti con il dataset filtrato, come discusso in 7.1, sono riportati nella tabella 7.3. La maggior parte degli estrattori ha restituito solo valori nulli, indicando la difficoltà di disambiguare in contesti specifici poco approfonditi nelle basi di conoscenza utilizzate.
I risultati del modello proposto sono visionabili in tabella 7.4.

Tabella 7.3: Risultati di GERBIL per il test set dei frammenti filtrato.

	F1 score	Precisione	Recall
AGDISTIS	0,01	0,01	0,01
AIDA	0	0	0
Babelfy	0	0	0
DBpediaSpotlight	0	0	0
FOX	0	0	0
PBOH	0,02	0,02	0,02

Tabella 7.4: Risultati del modello per il test set dei frammenti filtrato

	F1 score	Precisione	Recall
Modello proposto	0,14	0,08	0,78

7.4 Valutazione dei tweet di Twitter

Per quanto riguarda la valutazione del secondo dataset proveniente dal test set di Twitter, composto da 300 tweet, i risultati dell'esperimento effettuato su GERBIL sono riportati in tabella 7.5; inoltre sono visionabili anche all'indirizzo `http://gerbil.aksw.org/gerbil/experiment?id=201811260010`. La tabella 7.5 mo-

Tabella 7.5: Risultati di GERBIL per il test set dei tweet

	F1 score	Precisione	Recall
AGDISTIS	0,47	0,47	0,47
AIDA	0,41	0,79	0,28
Babelfy	0,38	0,75	0,25
DBpediaSpotlight	0,43	0,58	0,34
FOX	0,21	0,65	0,12
PBOH	0,52	0,53	0,52

stra dei risultati ben diversi dagli esisi in 7.1. Infatti come da previsioni, in ambito

generale e soprattutto in presenza di punteggiatura, che aiuta gli estrattori a dare un senso logico alla frase, i risultati ottenuti sono molto buoni. Discorso opposto per il modello creato, in quanto ritrovandosi in un contesto generale, i valori di similarità provenienti dal grafo di Wikidata e Wikipedia presentati in 5.3.4 ed utilizzati in 5.6 per disambiguare tra i diversi candidati di una menzione, non sono utilizzabili, perciò gli alberi di decisione hanno meno informazioni utili su cui basarsi. I risultati sono mostrati in 7.6. I valori di F-score ottenuti in questo contesto non

Tabella 7.6: Risultati del modello per il test set dei tweet

	F1 score	**Precisione**	**Recall**
Modello proposto	0,05	0,03	0,57

name	**MC**	**STMM**	**SLM**	**final score**	**Δ**
UniPI.3	0.561	0.474	0.456	0.5034	+1.27
UniPI.1	0.561	0.466	0.443	0.4971	+0.08
MicroNeel.base	0.530	0.472	**0.477**	0.4967	+0.10
UniPI.2	0.561	0.463	0.443	0.4962	+0.61
FBK-HLT-NLP.3	**0.585**	**0.516**	0.348	0.4932	+0.78
FBK-HLT-NLP.2	0.583	0.508	0.346	0.4894	+1.49
FBK-HLT-NLP.1	0.574	0.509	0.333	0.4822	+1.49
MicroNeel.merger	0.509	0.463	0.442	0.4751	+0.32
MicroNeel.all	0.506	0.460	0.444	0.4736	+38.56
sisinflab.1	0.358	0.282	0.380	0.3418	0.00
sisinflab.3	0.358	0.286	0.376	0.3418	+2.24
sisinflab.2	0.340	0.280	0.381	0.3343	+50.31
unimib.run_02	0.208	0.194	0.270	0.2224	+9.50
unimib.run_03	0.207	0.188	0.213	0.2031	+5.56
unimib.run_01	0.193	0.166	0.218	0.1924	0.00

Figura 7.2: Risultati challenge EVALITA (fonte: [32])

sono comparabili a quelli delle tecniche dello stato dell'arte, mentre il richiamo della tecnica proposta evidenzia una maggiore efficacia. Partendo da alti valori di recall, sarà possibile in futuro migliorare la parte di *disambiguation*, introducente tecniche aggiuntive, permettendo di generare risultati accettabili anche in contesti generali.
Per quanto riguarda i partecipanti, non si hanno a disposizione per tutti i contedenti i corrispettivi valori di precisione e recall, ma solo di F1-score finale, i quali vengono riportati in fig.7.2, nella colonna corrispondente a SLM (gli altri valori non sono utili per la valutazione). In fig.7.2 sono presenti 18 risultati diversi, perchè ogni gruppo partecipante poteva effettuare 3 sottomisioni diverse. Anche in questo caso i metodi utilizzati forniscono valori superiori al nostro modello.
Come ultima valutazione, è possibile fare un confronto con i valori di uno dei gruppi partecipanti ad EVALITA 2016, provenienti dal Politecnico di Bari, estrapolati dall'articolo da loro scritto, riportato in [44]. I risultati da loro ottenuti sono i seguenti:

- Recall = 0,28
- Precisione = 0,577
- F1 = 0,380

È possibile notare anche in questo caso, che il modello proposto riesce ad ottenere risultati più apprezabili in quanto a recall. La motivazione principale risiede nel fatto che il metodo proposto tiene in considerazione un gran numero di candidati effettuando la ricerca basata sulla similarità tra stringhe. Identificare un gran numero di candidati era un passo necessario in questo progetto, in quanto si era notato che gli estrattori in letteratura raggiungevano valori molto bassi di recall. Avere a disposizione più candidati comporta però uno sforzo maggiore nella fase di disambiguazione, infatti nei casi in cui le *features* disponibili non sono sufficienti a discriminare tra i candidati, la precisione risulta essere molto bassa (questo caso specifico si è verificato nella *disambiguation* dei candidati nel dataset di Twitter, in assenza dei due valori di similarità di Wikidata e Wikipedia).

Capitolo 8

Conclusioni e progetti futuri

Concludendo questo progetto, si è partiti con l'obiettivo di realizzare un modello di estrattore che fosse in grado di lavorare bene in situazioni critiche, come quelle presentate dai transcript ottenuti dal corso analizzato. I risultati sono stati soddisfacenti, in quanto confrontando i risultati in confronto agli estrattori presenti in letteratura, si sono ottenuti risultati decisamente migliori. La versione finale con cui si conclude questa parte di sperimentazione, è un primo prototipo di sistema che sarà effettivamente utilizzato un giorno come funzionalità aggiuntiva sul Portale della Didattica per gli studenti del Politecnico di Torino. Ora che è stato realizzato un primo modello, ed avendo una visione più generale rispetto ai mesi iniziali in cui non sempre è stato facile decidere il metodo migliore da adottare, sarà possibile effettuare dei miglioramenti mirati in ogni fase del modello. Esempio di miglioramenti possibili potrebbero essere:

- Cercare di ottenere un transcript migliore dal video, in modo tale da almeno riuscire ad individuare frasi logiche diverse;

- Creare una struttura più efficiente di quella utilizzata per accedere ai dati. In particolare si potrebbero creare più cluster, uno per ogni tipologia di label associata alle entità di cui disponiamo;

- Nella ricerca delle label più simili, sarebbe opportuno inserie nella equazione 5.3, anche un parametro che valuti la similarità tra i singoli token più vicini all'interno delle due stringhe. Ad esempio nei tweet capitava spesso che la stringa da cercare corrispondesse al nome di un personaggio famoso, ma su Wikidata la label più *vicina* corrisponde al nome seguito dal cognome. In questo esempio, aggiungendo questa misura di similarità, si riconoscerebbe alla perfezione il token del nome, perciò l'entità verrebbe inclusa tra i candidati;

- Sarebbe opportuno aumentare il numero di entità candidate prese in considerazione, in quanto in 5.5.3, dopo la fase di filtraggio, teniamo in considerazione tutte le entità con label corrispondenti con similarità uguale a 1 rispetto alla stringa campione, e le prime 10 entità con valori più vicini. Togliendo questo filtro, e prendendo tutte le entità con una label più simile di 0.7, si migliorerebbe certamente la scelta dei candidati, in quanto entità corrette che prima erano scartate, ora verrebbero tenute in considerazione, ma bisognerebbe risolvere il problema dell'alto numero di candidati;

- Ottenendo un transcript migliore e isolando frasi logiche diverse, si potrebbe valutare come parametro aggiuntivo la similarità tra la descrizione della frase contenente la menzione, e la descrizione dell'entità candidata.

- Sarebbe necessario trainare il modello utilizzato nell'albero di decisione con più corsi in quanto eseguendolo su un solo corso, il modello potrebbe risultare meno efficente;

- nella ricerca delle etichette candidate per una stringa campione, si potrebbe prendere in considerazione di cercare anche utilizzando la traduzione inglese della stringa, essendo le label inglesi più complete rispetto ad altre lingue.

- combinare i risultati finali, basandosi sia sui transcript, sia sulle slide.

Oltre a questi aspetti migliorabili, l'approccio presentato ha dimostrato alcune caratteristiche positive:

- ha dimostrato una buona capacità nel restituire valori alti di recall, più di ogni altro sistema presente in letteratura;

- utilizzando Wikidata come base di conoscenza, il nostro modello è indipendente dalla lingua, in quanto Wikidata non fa distinzione tra versioni diverse con lingue diverse come Wikipedia. Attualmente il modello è già utilizzabile anche per l'inglese, avendo preso in considerazione dal dump di Wikidata entrambe le lingue. Per applicarlo ad altre lingue sarebbe necessario prendere i valori della nuova lingua, aggiungerli alla struttura di clustering utilizzata, e il sistema dovrebbe teoricamente già funzionare. Ovviamente però i nostri risultati sono legati anche alla completezza di valori presenti su Wikidata, e non in tutte le lingue è così completa come per l'inglese e l'italiano;

- è stato in grado di dare risultati buoni anche in assenza di un input pulito.

Un passo finale inoltre, potrebbe essere quello di inserire questa nuova versione di estrattore all'interno di un approccio Ensemble, in particolare all'interno del progetto di Lorenzo Canale, dottorando che ha lavorato anche a questo progetto, utilizzando il metodo da lui descritto in [22]. Infatti, aggiungendo questo modello agli altri estrattori presenti nel suo sistema, ottenendo una recall superiore, si potrebbe migliorare ulteriormente i risultati finali.

Bibliografia

[1] P. G. Ipeirotis A. K. Elmagarmid e V. S. Verykios. "Duplicate Record Detection: A Survey". In: *IEEE Transactions on Knowledge and Data Engineering* 19 (2007), pp. 1–16.

[2] Alessandro Raganato Andrea Moro e Roberto Navigli. "Entity linking meets word sense disambiguation: a unified approach". In: *TACL* 2 (2014), pp. 231–244.

[3] J. Curran B. Hachey W. Radford. "Graph-based Named Entity Linking with Wikipedia". In: *Proc. of the 12th International Conference on Web Information System Engineering (Sydney, NSW, Australia, 2011)* (2011), pp. 213–226.

[4] R. Bunescu e M. Paşca. "Using encyclopedic knowledge for named entity disambiguation". In: *Proc. of the 11th Conference of EACL, pp. 9–16* (2006), pp. 9–16.

[5] P. Cimiano e J. Vlker. "Towards Large-scale, Open-domain and Ontology-based Named Entity Classification". In: *Proc. of Recent Advances in Natural Language Processing* (2005), pp. 166–177.

[6] S. Cucerzan. "Large-Scale Named Entity Disambiguation Based on Wikipedia data". In: *Proc. of EMNLP-CoNLL Joint Conference 2007)* (2007), pp. 708–716.

[7] R.L. Schwartz D.M. Bikel e R.M. Weischedel. "An Algorithm That Learns What's in a Name". In: *Machine Learning* 34 (1999), pp. 211–231.

[8] Gjergji Kasneci Fabian M. Suchanek e Gerhard Weikum. "Journal of Web Semantics". In: *Proceedings of the 13th International Semantic Web Conference (ISWC)* 6 (2008), pp. 203–217.

[9] M. Fleischman e E. Hovy. "Fine grained Classification of Named Entities". In: *Proc. of the 19th international conference on Computational linguistics* (2002), pp. 1–7.

[10] Markus Krötzsch Fredo Erxleben Michael Günther. "Introducing Wikidata to the Linked Data Web". In: *Proceedings of the 13th International Semantic Web Conference (ISWC)* (2014).

[11] Jim Giles. "Internet encyclopedias go head to head". In: *Nature 438* 7070 (2005), pp. 900–901.

[12] V. Tablan H. Cunningham K. Bontcheva. "GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications". In: *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics* (2002).

[13] R. Grishman H. Ji e H. T. Dang. “An Overview of the TAC2011 Knowledge Base Population Track”. In: *Proc.of Text Analysis Conference (TAC2011)* (2011).

[14] X. Han e L. Sun. “A Generative Entity-Mention Model for Linking Entities with Knowledge Base”. In: *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (Portland, Oregon, USA, June 19-24, 2011)* (2011), pp. 945–954.

[15] Hoffart. “Robust Disambiguation of Named Entities in Text”. In: *Natural Language Processing (Edinburgh, Scotland, UK, July 27–31, 2011)* (2011), pp. 782–792.

[16] Christian Bizer Jens Lehmann Robert Isele. “DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia”. In: *Semantic Web Journal, 2015* (2015).

[17] Max Jakob Joachim Daiber e Pablo N. Mendes. “Improving efficiency and accuracy in multilingual entity extraction”. In: *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)* (2013).

[18] Ilaria Bordino Johannes Hoffart Mohamed Amir Yosef. “Robust Disambiguation of Named Entities in Text”. In: *Conference on Empirical Methods in Natural Language Processing* (2011), pp. 782–792.

[19] J. Kleb e A. Abecker. “Entity Reference Resolution via Spreading Activation on RDF-Graphs”. In: *Proc. of the 2010 Extended Semantic Web Conference (ESWC 2010)* (2010).

[20] M. Anderson L. Ratinov D. Roth. “Local and Global Algorithms for Disambiguation to Wikipedia”. In: *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (Portland, Oregon, USA, June 19-24, 2011)* (2011), pp. 1375–1384.

[21] Dimosthenis Karatzas Lluis Gomez Marçal Rusinol. “LSDE: Levenshtein space deep embedding for query-by-string word spotting”. In: *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)* (2017), pp. 499–504.

[22] Pasquale Lisena Lorenzo Canale e Raphael Troncy. “A Novel Ensemble Method for Named Entity Recognition and Disambiguation based on Neural Network”. In: (2018).

[23] P. Ferragina M. Cornolti e M. Ciaramita. “A framework for benchmarking entity-annotation systems”. In: *22nd World Wide Web Conference* (2013).

[24] T. Finin M. Dredze P. McNamee. “Entity Disambiguation for Knowledge Base Population”. In: *Proc. of 23rd International Conference on Computational Linguistics (COLING 2010, Beijing, China, August 23-27, 2010)* (2010), pp. 23–27.

[25] R. Mihalcea. “Using Wikipedia for Automatic Word Sense Disambiguation”. In: *Human Language Technologies 2007: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT/-NAACL 2007, Rochester, New York, April 2007)* (2007).

[26] D. Milne e I.H. Witten. “Learning to Link with Wikipedia”. In: *Proc. of the 17 th ACM Conference on Information and Knowledge Management (CIKM 2008)* (2008), pp. 509–518.

[27] R. Navigli. “Word Sense Disambiguation: A Survey”. In: *ACM Computing Surveys* 41 (2008), pp. 1–69.

[28] Roberto Navigli e Simone Paolo Ponzetto. “ BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network”. In: *Artificial Intelligence* (2012).

[29] C. Legg O. Medelyan D. Milne. “Using Wikipedia for Automatic Word Sense Disambiguation”. In: *International Journal of Human-Computer Studies* 67 (), pp. 716–754.

[30] Thomas Hofmann Octavian-Eugen Ganea Aurelien Lucchi e L. Wesemann. “Probabilistic Bag-Of-Hyperlinks Model for Entity Linking”. In: *WWW’16* (2016).

[31] Max Jakob Pablo N. Mendes e Christian Bizer. “Dbpedia spotlight: Shedding light on the web of documents”. In: *Proceedings of the 7th International Conference on Semantic Systems, I-Semantics ’11* (2011), pp. 1–8.

[32] Anna Lisa Gentile Pierpaolo Basile Annalina Caputo e Giuseppe Rizzo. “Overview of the EVALITA 2016 Named Entity rEcognition and Linking in Italian Tweets (NEEL-IT) Task”. In: (2016).

[33] Goran Topić. Pontus Stenetorp Sampo Pyysalo. “brat: a web-based tool for NLP-assisted text annotation”. In: *Proceedings of the Demonstrations Session at EACL 2012* (2012).

[34] A. Csomai R. Mihalcea. “Wikify!: Linking Documents to Encyclopedic Knowledge”. In: *Proc. of the 16 th ACM Conference on Information and Knowledge Management (CIKM 2007)* (2007), pp. 233–242.

[35] G. Rizzo R. Usbeck M. Roder e L. Wesemann. “GERBIL – General Entity Annotation Benchmark Framework”. In: *24th WWW conference* (2015).

[36] Daniel Gerber Ricardo Usbeck e Andreas Both. “Agdistis - agnostic disambiguation of named entities using linked open data”. In: *International Semantic Web Conference* (2014).

[37] J. Lehmann S. Hellmann e M. Brummer. “Integrating NLP using Linked Data”. In: *12th International Semantic Web Conference* (2013).

[38] R. Rastogi S. Kataria K. Kumar. “Entity Disambiguation with Hierarchical Topic Models”. In: *Proc. of 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2011, August 21-24, 2011, San Diego, CA)* (2011), pp. 1037–1045.

[39] A. Singh S. Kulkarni e S. Chakrabarti. “Collective Annotation of Wikipedia Entities in Web Text”. In: *Proc. of the 15 th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD 2009)* (2009), pp. 457–466.

[40] E.F. Tjong Kim Sang e F. De Meulder. “Introduction tothe CoNLL-2003 Shared Task: Language Independent Named Entity Recognition”. In: *Proc. of CoNLL-2003* (2003), pp. 142–147.

[41] Nakatani Shuyo. “Language detection library for java”. In: (2010).

[42] Rene Speck e Axel-Cyrille Ngonga Ngomo. “Ensemble learning for named entity recognition”. In: *Proceedings of the International Semantic Web Conference, Lecture Notes in Computer Science* (2014).

[43] J. Hendler T. Berners-Lee e O. Lassila. “The Semantic Web”. In: *Scientific American* (2001), pp. 34–43.

[44] Tommaso Di Noia Vittoria Cozza Wanda La Bruna. “sisinflab: an ensemble of supervised and unsupervised strategies for the NEEL-IT challenge at Evalita 2016”. In: (2016).

[45] J. Su W. Zhang e C.-L. Tan. “A Wikipedia-LDA Model for Entity Linking with Batch Size Changing Instance Selection”. In: *Proc. of International Joint Conference for Natural Language Processing (IJCNLP 2011, Chiang Mai, Thailand, November 8-13, 2011)* (2011), pp. 562–570.

[46] J. Su W. Zhang e W. Wang. “Entity Linking Leveraging Automatically Generated Annotation”. In: *Proc. of 23rd International Conference on Computational Linguistics (COLING 2010, Beijing, China, August 23-27, 2010)* (2010), pp. 1290–1898.

[47] Y. C. Sim W. Zhang e C.-L. Tan. “Entity Linking with Effective Acronym Expansion, Instance Selection and Topic Modeling”. In: *Proc. of International Joint Conferences on Artificial Intelligence 2011 (IJCAI 2011, Barcelona, Spain, Jul 16-22, 2011)* (2011), pp. 1909–1904.

[48] L. Sun X. Han e J. Zhao. “Collective Entity Linking in Web Text: A Graph-Based Method”. In: *Proc. of the 34th Annual ACM SIGIR Conference (Beijing, China, July 24-28, 2011)* (2011), pp. 765–774.

[49] T. Liu Y. Guo W. Che. “A Graph-based Method for Entity Linking”. In: *Proc. of the 5th International Joint Conference on Natural Language Processing (IJCNLP 2011, Chiang Mai, Thailand, November 8-13, 2011)* (2011), pp. 1010–1018.

[50] L. Nie Y. Zhou e S. Gaffney. “Resolving Surface Forms to Wikipedia Topics”. In: *Proc. of 23rd International Conference on Computational Linguistics (COLING 2010, Beijing, China, August 23-27, 2010)* (2010), pp. 1335–1343.

[51] M. Huang Z. Zheng e X. Zhu. “Learning to Link Entities with Knowledge Base”. In: *Human Language Technologies 2010: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL 2010)* (2010).