POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea

Estrazione di entità da testi della Pubblica Amministrazione

Relatore

prof. Maurizio Morisio

Correlatore:

dott. Giuseppe Rizzo

Correlatore:

dott. Davide Allavena

Candidato

Davide Maietta

Dicembre 2021

Indice

Capitolo 1

Introduzione

L'avanzamento tecnologico delle telecomunicazioni negli ultimi decenni ha permesso alle persone di recuperare dati e informazioni provenienti dai luoghi più disparati e ad una velocità di approvvigionamento praticamente nulla: se prima di Internet l'unico modo di ricevere un documento ufficiale era farne richiesta all'ufficio competente, ora le documentazioni possono essere normalmente a disposizione in rete. Gli utenti della rete hanno a disposizione una mole così grande di documenti e informazioni che gli è praticamente impossibile visionare la totalità di questi dati. In questo contesto l'esigenza di estrarre dati da documenti è sempre più sentita, soprattutto quando si voglia fare una cernita dei documenti contenenti specifiche informazioni di proprio interesse. Questa tesi ha come scopo effettuare un'estrazione di dati da un insieme di documenti della Pubblica Amministrazione (PA); più precisamente i documenti appartengono al contesto dei bandi pubblici e i dati da cercare sono tutte quelle informazioni che impongono dei requisiti ai possibili partecipanti di gara. Il lavoro è strutturato come segue: al capitolo secondo si descrivono i bandi pubblici e dati SOA in essi contenuti; al capitolo terzo si presenta il problema della ricerca di tali dati formalizzata come un problema di Named Entity Extraction. Al capitolo quarto viene svolta una rassegna di lavori correlati, che vertono sulle metodologie di estrazione di dati; dopodiché tali metodologie vengono implementate al capitolo quinto. In questo contesto si provano approcci tecnologici diversi, nella fattispecie l'uso delle regular expressions e delle reti neurali. Di entrambe le tecnologie vengono delineati punti di forza e di debolezza, per poi investigare se sia preferibile l'una delle due o se invece sia possibile un compromesso che valorizzi il meglio di entrambe. Al capitolo sesto si introduce un approccio ibrido configurato come Human In The Loop, che viene proposto come soluzione vantaggiosa rispetto alle tecnologie singole considerate inizialmente. Nell'ultimo capitolo sono esposte le conclusioni e le possibilità di sviluppi successivi per questo lavoro di ricerca.

Capitolo 2

Estrazione di dati da testi della pubblica amministrazione

Tra le funzioni di uno Stato vi sono la costruzione, la manutenzione e la messa in sicurezza delle infrastrutture di pubblica utilità, quali ad esempio gli edifici pubblici, le reti di telecomunicazione, le strade e gli ospedali di cui beneficiano i cittadini. La Pubblica Amministrazione, dovendo garantire una quantità così grande di infrastrutture, non esegue i lavori direttamente, ma ne delega l'attuazione pratica a delle imprese private scelte con una pubblica gara d'appalto. Nel presente capitolo partiamo dallo scenario degli appalti per introdurre i dati SOA, che saranno oggetto della nostra ricerca.

2.1 Gare d'appalto

L'appalto è un contratto con il quale una parte, detta parte appaltatrice, assume, "con organizzazione dei mezzi necessari e con gestione a proprio rischio, l'obbligazione di compiere in favore di un'altra (committente o appaltante) un'opera o un servizio"[1]. Gli appalti pubblici sono dunque un modo per eseguire delle opere pubbliche pagando delle imprese private per la realizzazione vera e propria; la PA deve però assicurarsi che: l'impresa abbia le conoscenze per portare a termine l'opera rispettando degli standard tecnici, ossia seguendo la **regola d'arte**; l'impresa gestisca i lavori in maniera competitiva, senza comportare sprechi ingiustificati di soldi pubblici; l'impresa riesca a reggere lo sforzo finanziario che l'opera comporta, sia per quanto riguarda l'approvvigionamento dei materiali, sia per quanto riguarda il costo del lavoro. L'assegnazione di un progetto si avvale di apposita gara d'appalto, che ha come obbiettivo la selezione dell'impresa che meglio possa soddisfare questi requisiti di competenze tecniche, finanziarie e di competitività sul mercato.

2.2 Identificativi di progetto e di gara

Nell'ambito di una gara d'appalto, il progetto può essere composto da più parti singole chiamate lotti; l'assegnazione di un lotto è indipendente dagli altri, per cui diversi lotti di un progetto possono essere assegnati ad aziende differenti nell'ambito della stessa gara d'appalto. I documenti che andiamo ad analizzare riportano una serie di informazioni di nostro interesse, quali:

- il **Codice Unico di Progetto** (**CUP**), che descrive univocamente il progetto d'investimento pubblico; è un identificativo alfanumerico di quindici caratteri;
- il **Codice Identificativo di Gara** (**CIG**), che indica in maniera univoca il lotto; è un identificativo alfanumerico di dieci caratteri;

2.3 Lavori e importi

La gara d'appalto deve indicare in maniera evidente e inoppugnabile quali competenze tecniche e quali capacità finanziarie sono ritenute requisiti fondamentali per la partecipazione; qualsiasi ambiguità nei requisiti di una gara potrebbe dare adito a ricorsi e quindi a rallentare la gara stessa. Il bisogno di requisiti oggettivi ha motivato l'introduzione di apposite certificazioni fornite dalle Società Organismi di Attestazione (SOA), che prendono il nome di "certificati SOA" o "attestazioni SOA"[3]. Tali attestazioni si dividono in due macrogruppi:

- **Categorie di opere:** individuano le categorie di lavori dal punto di vista tecnico;
- **Classifiche di importi:** individuano i livelli di capacità finanziaria.

Un bando di gara esprimerà i requisiti di progetto sotto forma di categorie e classifiche SOA, per cui un'azienda che voglia essere ammessa alla gara dovrà possedere le attestazioni SOA richieste.

2.4 Tassonomia SOA

Le Categorie sono suddivise in due macro-categorie: opere generali e opere specializzate, rispettivamente identificate dagli acronimi "OG" e "OS"; sono individuate 13 categorie di Opere Generali e 39 categorie di Opere Specializzate. Le Classifiche d'importo, in numero di dieci, sono rese come numeri ordinali romani.

2.4.1 Categorie di opere generali

- OG-1, Edifici civili e industriali
- OG-2, Restauro e manutenzione dei beni immobili sottoposti a tutela
- OG-3, Strade, autostrade, ponti, viadotti, ferrovie, metropolitane
- OG-4, Opere d'arte nel sottosuolo
- OG-5, Dighe
- OG-6, Acquedotti, gasdotti, oleodotti, opere di irrigazione e di evacuazione
- OG-7, Opere marittime e lavori di dragaggio
- OG-8, Opere fluviali, di difesa, di sistemazione idraulica e di bonifica
- OG-9, Impianti per la produzione di energia elettrica
- OG-10, Impianti per la trasformazione alta/media tensione e per la distribuzione di energia elettrica in corrente alternata e continua ed impianti di pubblica illuminazione
- OG-11, Impianti tecnologici
- OG-12, Opere ed impianti di bonifica e protezione ambientale
- OG-13, Opere di ingegneria naturalistica

2.4.2 Categorie di opere specializzate

- OS-1, Lavori in terra
- OS-2-A, Superfici decorate di beni immobili del patrimonio culturale e beni culturali mobili di interesse storico, artistico, archeologico ed etnoantropologico
- OS-2-B, Beni culturali mobili di interesse archivistico e librario
- OS-3, Impianti idrico-sanitario, cucine, lavanderie
- OS-4, Impianti elettromeccanici trasportatori
- OS-5, Impianti pneumatici e antintrusione
- OS-6, Finiture di opere generali in materiali lignei, plastici, metallici e vetrosi
- OS-7, Finiture di opere generali di natura edile e tecnica
- OS-8, Opere di impermeabilizzazione

- OS-9, Impianti per la segnaletica luminosa e la sicurezza del traffico
- OS-10, Segnaletica stradale non luminosa
- OS-11, Apparecchiature strutturali speciali
- OS-12-A, Barriere stradali di sicurezza
- OS-12-B, Barriere paramassi, fermaneve e simili
- OS-13, Strutture prefabbricate in cemento armato
- OS-14, Impianti di smaltimento e recupero rifiuti
- OS-15, Pulizia di acque marine, lacustri, fluviali
- OS-16, Impianti per centrali produzione energia elettrica
- OS-17, Linee telefoniche ed impianti di telefonia
- OS-18-A, Componenti strutturali in acciaio
- OS-18-B, Componenti per facciate continue
- OS-19, Impianti di reti di telecomunicazione e di trasmissioni e trattamento
- OS-20-A, Rilevamenti topografici
- OS-20-B, Indagini geognostiche
- OS-21, Opere strutturali speciali
- OS-22, Impianti di potabilizzazione e depurazione
- OS-23, Demolizione di opere
- OS-24, Verde e arredo urbano
- OS-25, Scavi archeologici
- OS-26, Pavimentazioni e sovrastrutture speciali
- OS-27, Impianti per la trazione elettrica
- OS-28, Impianti termici e di condizionamento
- OS-29, Armamento ferroviario
- OS-30, Impianti interni elettrici, telefonici, radiotelefonici e televisivi
- OS-31, Impianti per la mobilitÃ sospesa
- OS-32, Strutture in legno
- OS-33, Coperture speciali
- OS-34, Sistemi antirumore per infrastrutture di mobilitÃ
- OS-35, Interventi a basso impatto ambientale

2.4.3 Classifiche di importi

- I classifica, fino a euro 258.000
- II classifica, fino a euro 516.000
- III classifica, fino a euro 1.033.000
- III bis classifica, fino a euro 1.500.000
- IV classifica, fino a euro 2.582.000
- IV bis classifica, fino a euro 3.500.000
- V classifica, fino a euro 5.165.000
- VI classifica, fino a euro 10.329.000
- VII classifica, fino a euro 15.494.000
- VIII classifica, oltre euro 15.494.000

Capitolo 3

Casi d'uso delle attestazioni SOA

Finora sono state descritte le attestazioni SOA ed è stato chiarito il ruolo fondamentale che queste certificazioni ricoprono negli appalti. D'ora in avanti si intende descrivere l'uso di queste attestazioni da parte dei vari attori del mondo degli appalti:

1. la **Pubblica Amministrazione**: produce il bando di gara, relativo ad un progetto identificato univocamente dal codice CUP ed eventualmente diviso in lotti, ognuno identificato da un codice CIG; nel bando di gara aggiunge i requisiti di Categorie SOA e le Classifiche economiche SOA, imponendo eventualmente vincoli temporali su tali certificati ("L'azienda deve risultare in possesso di tale certificato dal 2020"); il bando viene scritto in formato pdf e pubblicato sul sito dell'ente pubblico (Comune, Provincia, Regione, Ministero, et cetera), che risulta in questo contesto essere l'appaltante;

2. l'**imprenditore**, o qualsivoglia candidato appaltatore: naviga i siti web della PA alla ricerca di bandi di gara; per ogni bando di gara trovato, deve comprendere i requisiti SOA e accertarsi di poterli soddisfare; solo se dotato di idonea attestazione, potrà candidarsi ad essere appaltatore prendendo parte alla gara d'appalto.

Sia chiaro che la P.A. pubblica anche altre tipologie di documenti, per cui è utile precisare che ci riferiamo a quell'insieme di documenti che descrivono gare e verbali d'appalto: possiamo riferirci a questo insieme chiamandolo dominio d'appalto. In questi due casi d'uso possiamo evidenziare una prima problematica: ogni ente pubblico ha un proprio sito web e la raccolta di documenti di appalti può essere un'attività lunga e dispersiva. Questo problema potrebbe essere risolto da un software di tipo web-crawler che navighi i siti web della PA e raccolga tutti i documenti del dominio d'appalto in un dataset. Esiste una seconda problematica da porre: ammesso che abbia a disposizione tutto il dataset del dominio degli appalti, l'imprenditore che è alla ricerca di un bando adatto alla sua specifica certificazione si troverà costretto a verificare per ogni documento se tale certificazione sia sufficiente per essere ammesso al bando; detto in altre parole, dovrà leggere e annotare

manualmente tutti i documenti in base alle attestazioni SOA per poi scegliere il bando con l'annotazione SOA desiderata. Questa problematica è decisamente noiosa, sia perché la lettura toglie alla persona un ingente tempo di lavoro, sia perché un compito del genere è perfettamente automatizzabile; d'ora in poi ci focalizzeremo su questa attività specifica. L'idea di base è permettere alle imprese di migliorare la ricerca di possibili appalti rendendo questi documenti digitalmente navigabili in base alle attestazioni SOA: per l'imprenditore il caso d'uso 2 si ridurrebbe a cercare l'attestazione SOA desiderata per avere tutti e soli i bandi che la contemplino tra i requisiti. La ricerca di attestazioni SOA in un testo è un problema di **Named Entity Recognition** (**NER**).

Formulazione del problema: *Dato un insieme di documenti in formato testuale appartenenti al dominio delle gare d'appalto, estrarre le attestazioni SOA ivi menzionate.*

3.1 Valori testuali ricercati

L'impostazione del problema inizia con la definizione degli insiemi di valori possibili; nel caso delle attestazioni SOA, seguendo la tassonomia esposta definiamo l'insieme delle Categorie e l'insieme delle Classifiche...:

Categorie = {"OG-1","OG-2", ..., "OG-13", "OS-1","OS-2", ..., "OS-35"}

Classifiche = {"I", "II", "III", "IV", "IV-bis", "V", "VI", "VII", "VIII"}

In definitiva, il sistema indicherà le informazioni estratte in base all'insieme Categorie e Classifiche così definiti.

Formato dei dati estratti: Più in generale, per ogni attestazione SOA che il sistema individuerà in un documento, dovranno essere mandate in output le seguenti informazioni:

- la porzione di testo in cui l'attestazione SOA è individuata;
- la categoria riconosciuta;
- la classifica riconosciuta;
- lo score, ossia il grado di accuratezza dell'output.

soa_extracted_tuple = [(*start_offset_cat*, *end_offset_cat*), (*cat*, *class*), *score*)]

3.2 Dominio del modello

Una volta considerati i dati in questione, dobbiamo articolarli in un modello di dominio in cui abbia senso operare. Solitamente i modelli NER contemplano entità come "Organization", "Location", "Person", ognuna delle quali copre molte istanze testuali con informazioni diverse. Ad esempio, nelle seguenti frasi:

- "L'ONU si riunirà per discutere dell'emergenza climatica mondiale."
- "La Caritas metterà a disposizione ulteriori posti letto per i senzatetto."
- "La FIGC agisce attraverso i comitati regionali e le delegazioni provinciali e locali."

le diciture "L'ONU", "La Caritas" e "La FIGC" sono entità "Organization".

Nel nostro specifico caso abbiamo due entità che possiamo chiamare "CATEGORIA-SOA" e "CLASSIFICA-SOA", ma al loro interno dobbiamo fare delle opportune distinzioni: tra diversi tipi di "CATEGORIA-SOA" e tra diversi tipi di "CLASSIFICA-SOA". Questo ci impone di contemplare nel **dominio** non solo l'entità, ma anche il tipo, che vincola il sistema NER a riconoscere una specifica gamma di dati.

Definiamo come segue le entities e gli entity type:

- entity "CATEGORIA-SOA", comprendente gli entity type "OG-1" , . . . , "OG-13", "OS-1", . . . , "OS-35" come definiti al 2.4.1;
- entity "CLASSIFICA-SOA", comprendente gli entity type "I", "II" , . . . , "VIII" come definiti al 2.4.3.

3.3 Valutazione dei dati estratti

Quando il sistema collega la stringa "o.g. 1" all'elemento OG-1, sta effettuando una classificazione: sta classificando un dato come appartenente alla classe delle istanze di OG-1. Il sistema di estrazione dati effettua una classificazione su ogni porzione di testo individuata; ogni porzione di testo può essere collegata alla classe di uno dei valori SOA. Tipicamente, in un problema di classificazione l'output può essere valutato come segue:

- True Positive (TP), se l'elemento è stato assegnato correttamente alla classe;
- False Positive (FP), se l'elemento è stato assegnato erroneamente alla classe;
- True Negative (TN), se l'elemento non è stato assegnato alla classe perché non vi andava assegnato;

- False Negative (FN), se l'elemento non è stato assegnato alla classe ma vi andava assegnato.

Ogni output del sistema può essere valutato in termini di TP-FP-TN-FN; disponendo del numero di TP-FP-TN-FN è poi possibile calcolare alcune metriche che descrivono le performance del sistema:

Precision = (TP)/(TP + FP)

Recall = (TP)/(TP + FN)

Accuracy = (TP + TN)/(TP + TN + FP + FN)

L'accuratezza di un sistema è spesso calcolata con la *F1-score*, costituita dalla media armonica di Precision e Recall:

F1 = (2*P*R)/(P+R)

3.4 Ground truth

Per ogni istanza di testo assegnato ad una classe, bisogna affermare se la classificazione data costituisce true positive, false positive, true negative o false negative; questo vuol dire che, per valutare la bontà di una classificazione, abbiamo bisogno di stabilire quale sia la classificazione giusta. Il sistema che stiamo considerando effettua un'estrazione di dati, classificandoli come elementi delle attestazioni SOA; per valutare la bontà del sistema abbiamo dunque bisogno di stabilire per ogni porzione di testo l'output giusto, ovvero di definire la **verità** in base alla quale valutare il sistema. La **Ground Truth** è dunque l'output corretto che ci si aspetterebbe dal sistema; e come tale indicherà, per ogni porzione di documento, l'attestazione "giusta" che il sistema avrebbe dovuto estrarre. Un sistema di apprendimento dotato di Ground Truth appartiene all'approccio detto **Apprendimento Supervisionato**. Disponendo di una ground truth, potremo procedere a valutare l'output del sistema e quindi classificare ogni entità in output come true positive, false positive, true negative, false negative. Una volta contrassegnata ogni tupla dell'output come TP, FP, TN, FN, sarà possibile calcolare Precision (P), Recall (R) e Accuracy (A).

3.5 Tecniche di estrazione

Di seguito si espongono due approcci principali per l'estrazione di dati:

- l'uso di regular expression;

- l'uso di tecniche di machine learning.

Mostreremo che hanno differenti punti di forza, differenti punti di debolezza e confronteremo i risultati di entrambe le tecnologie.

Capitolo 4

Estrazione di entità da testi

In questo capitolo vogliamo focalizzarci sulla Named Entity Recognition (NER) e sulle varie possibilità di estrarre e classificare informazioni contenute in documenti di testo. L'estrazione di pattern testuali è una pratica consolidata nella programmazione tradizionale che fa abbondante uso delle **regular expression**; tuttavia, le regex soffrono di una serie di problematiche che le rendono uno strumento poco affidabile se usato da solo. Tali problematiche possono essere arginate e risolte se alle regex si affiancano altre tecnologie. Un altro strumento usato in NER sono le reti neurali o Neural Network (NN), che a differenza delle regex non "apprendono" un pattern testuale con una grammatica regolare, ma con l'uso di un quantitativo massiccio di esempi di dati **di training** opportunamente classificati, con un approccio che in Machine Learning viene definito **supervisionato**. Vedremo che la Named Entity Recognition può avvalersi anche di una tecnologia ibrida che sfrutti in contemporanea strumenti diversi, tra cui regular expressions, ontologie e reti neurali. Infine un pattern degno di nota è lo Human In The Loop (HITL), che permette l'intervento di un revisore umano per apportare correzioni alle entità estratte dal sistema NER; tale pattern si rivela particolarmente vantaggioso perché abilita il sistema NER ad **apprendere** le correzioni apportate per migliorare le prestazioni successive.

4.1 NER con Regex

Le regex sono notoriamente un potente strumento software, ma risultano difficilmente leggibili, i loro usi pratici sono documentati poco e si rivelano poco manutenibili; la comunità dei programmatori ha persino coniato il detto *"Now you have two problems"*[4], che esprime come le soluzioni software basate su regex, lungi dall'essere considerate affidabili, creino ulteriori problemi a causa della gestione delle regex stesse. Nello studio *"How to invest my time"* [10] gli autori, ben consapevoli di quanto le regex siano complesse ed error-prone, si domandano *fino a che punto* possano essere usate vantaggiosamente; più in particolare il loro studio si cala nel

contesto della Entity Extraction e si pone l'obiettivo di usare al meglio le risorse umane, studiando due attività diverse e complementari:

1. lo sviluppo di regex per produrre automaticamente annotazioni dati testuali;
2. l'annotazione manuale delle entità contenute nei dati stessi.

Le due attività sono compiute da operatori **umani**, per cui gli autori con questo studio hanno voluto indagare come utilizzare al meglio il tempo di un dipendente annotatore e programmatore, provando varie combinazioni delle due attività menzionate. La regex prodotta al punto 1 genera annotazioni dette **weak labels**, che vanno a costituire il **training set** della Neural Network. Al punto 2 l'annotatore può creare annotazioni *ex novo* per ampliare il dataset di training, ma può anche effettuare attività di **fine tuning**, correggendo le weak labels che la regex ha generato. Le due azioni concorrono a formare, addestrare e infine perfezionare una Neural Network e sono state sperimentate con differenti modalità temporali, creando scenari in cui il tempo è stato speso in diverse proporzioni sulla prima o sulla seconda attività. I risultati sperimentali di questo studio mostrano che:

- se il tempo da investire nella EE è poco (inferiore ai 40 minuti), conviene che l'operatore si limiti a produrre una regex;
- se il tempo è molto (superiore ai 40 minuti), l'operatore potrebbe spendere tutto il tempo a sua disposizione per creare annotazioni con cui istruire la rete neurale;
- tra i due casi estremi, può convenire che l'operatore umano spenda pochi minuti per creare una regex per un primo setup di rete neurale, per poi aggiungervi annotazioni manuali per farne fine-tuning.

Questo approccio che contempla le azioni umane in un sistema automatico da addestrare e perfezionare è detto **Human In The Loop**.

4.2 Deep Learning con Human-In-The-Loop

Nel campo della Named Entity Recognition(NER) i metodi di Deep Learning hanno un discreto successo, perché richiedono un'ingegnerizzazione limitata [9]; allo stesso tempo però hanno bisogno di grandi quantità di dati per effettuare il training dei modelli. Lo studio Improving Named Entity Recognition propone quindi un uso iterativo degli annotatori umani all'interno del sistema Human NERD (dove NERD sta per Named Entity Recognition with Deep Learning). Questo sistema di **Human In The Loop** si articola così:

1. Viene raccolto un dataset T di documenti non annotati;

2. Al sistema Human NERD vengono forniti modelli NER esterni da acquisire; importando tali modelli, Human NERD effettua una prima annotazione dei documenti del dataset T;

3. Human NERD invia ciascun documento preannotato ad un annotatore umano; poiché si prevede che gli annotatori possano essere in numero maggiore di uno, ogni documento apparterrà esclusivamente ad un annotatore; l'annotatore prescelto effettua la review delle annotazioni, aggiungendo, rimuovendo o correggendo etichette. Il risultato della revisione viene inviato al framework;

4. Basandosi sulle correzioni ricevute, Human NERD può aggiornare il modello in maniera incrementale; in alternativa può fare training da zero, istruendo così un modello nuovo.

5. Basandosi sui cambiamenti effettuati, Human NERD calcola il nuovo livello di Accuracy, computa il numero di occorrenze per classe d'entità, calcola il loss basato sulle attività di training e labelling; infine calcola una stima del gain dovuta al miglioramento dell'accuracy.

In definitiva l'approccio **Human In The Loop** è senz'altro promettente per la costruzione di dataset e per il training di modelli NER sempre più accurati.

4.3 Liste di dati da documenti sottoposti a OCR

Altro interessante contributo al Natural Language Processing è lo studio di Packer et al.[8], in cui si propone il funzionamento del software ListReader per l'acquisizione di dati da documenti sottoposti ad OCR. Più nello specifico ListReader acquisisce **liste** di dati; spetta ad un utente utilizzare l'interfaccia grafica di ListReader per compilare un generico form che descriva la struttura dei dati: ad esempio, si pensi ad una lista di persone; ogni elemento della lista dovrà fornire il Nome e il Cognome della persona ed eventualmente altre informazioni anagrafiche; l'utente che osserverà tali dati li userà per creare un form contenente i textbox "Nome", "Cognome", "Data di nascita", "Data di Decesso", etc. Da questa parziale annotazione ListReader potrà:

1. popolare un'ontologia con entità e attributi derivanti dal form;

2. indurre una regex per ogni dato identificato, costruendo un wrapper di regex iniziali;

3. generalizzare le regex iniziali con un algoritmo A*, per renderle adeguate a catturare anche dati più complessi;

4. fare **active learning**: consentire all'utente di modificare il form iniziale quando si presentino dati non aderenti alla struttura descritta dal form.

Infine ListReader risulta più performante degli algoritmi CRF per l'acquisizione di elementi da liste.

Capitolo 5

4. Realizzazione dei prototipi

In questa sezione andremo a illustrare come sono stati realizzati i due approcci di Named Entity Recognition. Un primo prototipo è stato realizzato infatti con le tradizionali tecniche di programmazione e con l'uso di espressioni regolari. Il secondo prototipo invece è stato realizzato con tecniche di Machine Learning.

5.1 Espressioni regolari

Il prototipo in questione ha come obiettivo la ricerca di attestazioni SOA all'interno di un testo. Nello specifico, il prototipo adopera tecniche di programmazione tradizionale, usando degli script Python ed effettuando ricerche di stringhe con espressioni regolari

Formulazione del problema: *Dato un insieme di documenti in formato testuale appartenenti al dominio delle gare d'appalto, estrarre le attestazioni SOA ivi menzionate.*

5.2 Accesso ai documenti del dataset

Il dataset dei documenti di gare d'appalto è implementato con un file csv. Tale file dunque descrive ogni documento come campi comma-separated; alcuni di questi campi sono l'id del documento, il testo del documento, il cig del bando, lo score della classificazione. Per l'acquisizione di dati si è adoperata la libreria Pandas: questa consente di convertire i dati testuali comma-separated in un apposito oggetto Pandas Dataframe.

```
dataframe = pd.read_csv(filepath, sep=',')
```

Una volta importato il csv in un dataframe, accediamo al dataframe esattamente come se fosse una matrice; di seguito il codice per stampare il primo elemento del campo testo:

```
print(dataframe['testo'][0])
```

Una volta estratti gli elementi di testo del dataframe, possiamo applicare le espressioni regolari alla ricerca delle attestazioni SOA su ogni singolo testo.

5.3 Raccolta e filtraggio di istanze

L'individuazione e la classificazione di attestazioni SOA da un testo richiedono due azioni:

1. Raccolta delle istanze: trovare una porzione di testo che abbia la struttura di un valore SOA;
2. Filtraggio delle istanze: verificare che quel dato corrisponda ad una attestazione SOA valida.

Per dare un esempio, posso coprire la prima azione con una regex minimale che descriva la categoria come "OS" seguito da due cifre:

```
regex_categoria = r'O(S|G)(\d\d?)'
```

Ciò non ci assicura di avere una categoria valida: ad esempio OS99 fa match con la regex, ma non è una categoria valida, poiché le opere specialistiche si fermano alla OS35. Similmente, le opere generali si fermano alla OG13; la regex banale mostrata in quest'ipotesi cattura benissimo un'istanza di testo come "OG14", che però non esiste nell'insieme delle categorie. La seconda azione consiste quindi nel verificare l'esistenza della stringa nell'insieme Categorie e in tal caso classificarla come tale.

L'ultimo punto da introdurre è la normalizzazione delle istanze individuate: è infatti possibile che l'entità "Opera Specialistica numero 1" abbia in pratica scritture diverse, come "OS1", "O.S.1", "O.S.-1" e così via, a seconda dello stile di scrittura di chi ha redatto un documento. Per quanto una buona regex possa raccogliere tutte queste istanze e pur essendo tutte queste istanze valide, il prototipo in questione darà ai dati individuati una forma **normalizzata** ovvero standardizzata.

Infine la procedura svolta è descrivibile a grandi linee come segue:

1. Nei punti di interesse, si individua una porzione di testo o *finestra*, all'interno della quale si cercano le attestazioni;
2. internamente alla finestra si applicano le espressioni regolari relative a categoria SOA e classifica SOA;
3. i risultati trovati vengono posti in una forma standard o *normalizzata* e filtrati rispetto al dizionario dei valori SOA.

5.4 Ricerca e classificazione delle attestazioni SOA

Analizzando un documento testuale, possiamo con poche regex raccogliere la maggior parte dei dati di nostro interesse. I dati SOA precedentemente illustrati hanno infatti una certa regolarità nei nomi: per raccogliere le categorie, ci basterà trovare tutte le parole aventi per prefissi "OS" e "OG"; non tutte le parole contenenti questi due prefissi sono automaticamente dati SOA, ma partiamo da questi match per isolare delle porzioni di testo che costituiranno le *finestre d'interesse* da verificare. In tali finestre andremo a verificare che i dati descrivano effettivamente delle sigle SOA e andremo a cercare le classifiche economiche. Schematizziamo di seguito il funzionamento dell'estrazione dati e classificazione con regex.

L'output di questo prototipo-regex mostra delle limitazioni e dei casi tipici di errore: in alcune istanze testuali può capitare che una categoria sia menzionata con un nome leggermente diverso da quello canonico (ad es. "OS-18", laddove le alternative accettabili sono "OS-18A" e "OS-18B"). In questi casi il filtering, la normalizzazione e anche le regex possono essere leggermente modificate per riconoscere come accettabili tali valori testuali; però modifiche ripetute alle regex possono portare le stesse a diventare incomprensibili e ad essere error-prone.

In definitiva, le regex hanno il vantaggio di produrre dei risultati immediati, ma non molto adatti a testi troppo variegati. Di seguito forniamo una visione più dettagliata sulle regex usate relativamente alle categorie e alle classifiche economiche.

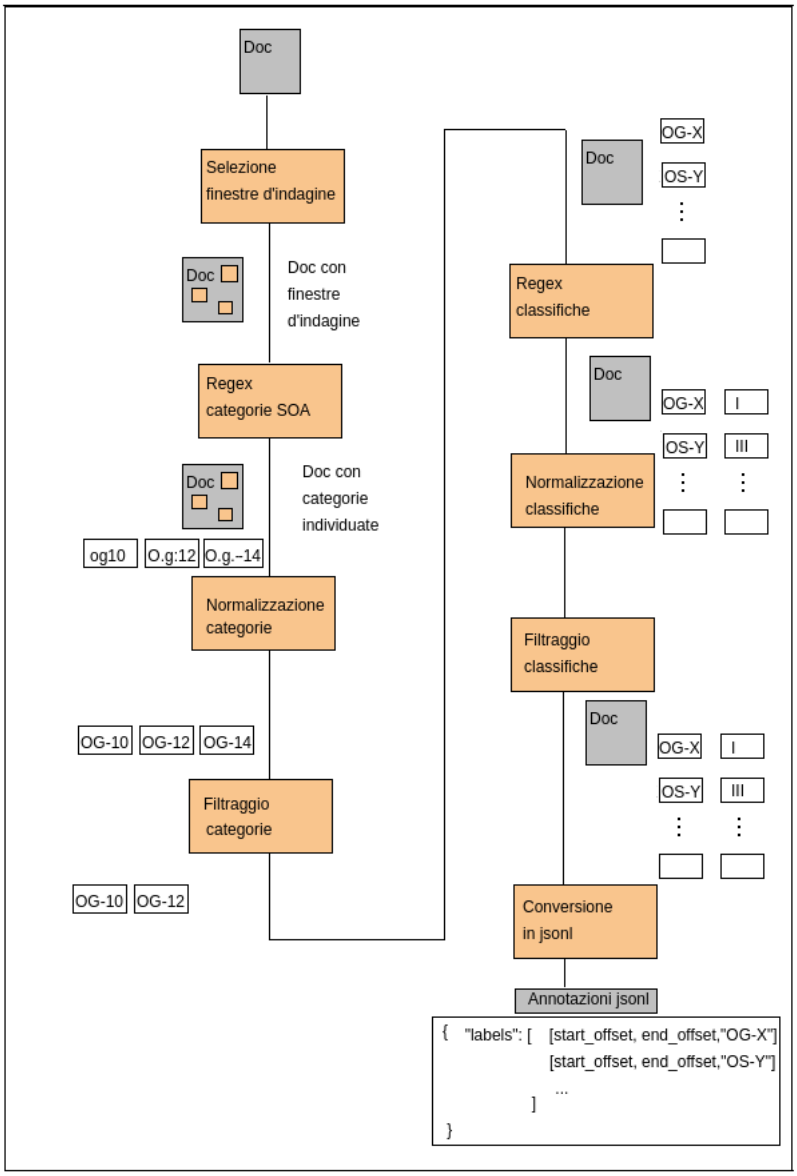

Figura 5.1: Human In The Loop framework

5.5 Finestre, espressioni regolari, filtering

La Finestra di caratteri è selezionata nel modo seguente: la presenza di una delle stringhe { “OS”, “Os”, “OG”, “Og” } costituisce l’inizio della finestra; successivamente vengono selezionati 100 caratteri. Regex adoperata:

```
from_os_n_char = r'O(S|G|s|g).{1,100}'
```

Ricerca delle categorie SOA operata internamente alla finestra: la presenza di una delle stringhe { “OS”, “Os”, “OG”, “Og” } costituisce l’inizio della presunta

attestazione; la stringa deve continuare con un separatore e al più due cifre decimali (ad es. “Os-98”, “Os-5”) la stringa può continuare con una lettera, che ci si aspetta essere la sottocategoria { “A”, “B” }; ad esempio è accettabile una stringa del tipo “OS-18A”. Con le dovute cautele sui vari tipi di carattere separatore {“-”,“.”, } e volendo tollerare eventuali caratteri di andata a capo, ottengo l’espressione regolare:

```
cat_regex = r’O(S|G|s|g)\n?[-\s]?\n?(\d\d?\w?)’
```

Normalizzazione delle categorie: Viene svolta con regex-substitution; seleziona la dicitura { “OS”, “Os”, “OG”, “Og” } e il gruppo (dd?w?) costituito dal numero e dalla sottocategoria “a”/“b”. Ad esempio: le istanze

```
"OS 10" , "OS<tab>10" e "OS\n10"
```

con la normalizzazione assumono la stessa forma “OS-10”.

Filtering: seleziona le attestazioni *sensate* in quanto contenute nel relativo dizionario. È infatti possibile che un’attestazione sia accettabile per la regex (ad es. OS-77) ma non abbia senso nel dominio delle categorie SOA.

Ricerca di classifiche economiche: numeri romani; mi assicuro che le lettere siano effettivamente numeri piuttosto che inizi di parole (e.g. la “I” maiuscola in “OS-12 Importo uguale a ” non va considerata un match con la classifica I) Regex adoperata:

```
class_regex =’(IV bis|IV|III|II|IX|VIII|VII|VI|X|V|I)[^\w]’
```

5.6 Errori comuni riportati

Dovendo valutare l’output del prototipo, raccolgo qui gli errori raggruppandoli in alcune categorie principali. Dalla tabella contenuta nel foglio di calcolo condiviso si possono vedere le corrispondenze tra input e output, compresi i casi di errore. Per questo report stati considerati i primi 100 elementi della tabella soa.csv

Gli errori riportati sorgono in maniere diverse: alcuni nascono a livello di regex, alcuni casi sono causati dallo script e dal dizionario fissato nello script; altri ancora dipendono da particolarità del documento.

Categoria non riconosciuta:

```
riga   657,  OS "I
riga 1181, OGI I si ? 105.541,05 | 62,901
riga 790, OG1IV bis e categoria scorporabile a;
riga 792, OG1III e categoria scorporabile a;

riga 130, "OG  11 di cui";
riga 540, OS   21; altre simili: 541,542
riga 135, "OS 12-A, OS 18-A, OS 21  e  OS  2-A"; altre: 382,487, 1159, 1160,1
riga 143, "OS.2"; altre: 145, 148, 254;
```

Motivazioni:

1. caratteri estranei a quelli previsti dalla regex;
2. spaziature non previste nella soa_cat_regex;
3. nella cat_regex il match avviene correttamente solo con "OS-12A"; non è previsto il carattere "-" prima della lettera "A";
4. nella cat_regex non è previsto il carattere di interpunzione "."

Classifica non acquisita:

```
riga 61, "OS21 nella classe I.";
riga 124,"OG 3 classifica II;";
riga 371, "OG 3 classifica I;";
riga 416, OG 12 III;
riga 1251, OG 3 classifica V;
riga 1262, OG1 Classifica I;
riga 825,  "OG1 V OS3 VE OS28IV BIS"
```

Motivazioni:

1. errore lato script: lo script seleziona una sottostringa con il carattere finale mancante, causando il mancato match con la class_regex;
2. "OS3 VE" (dove VE indica "V classifica Economica") non è previsto dalla regex della classifica.

Sottoclassifica non acquisita: riga 125, "OS21 —IIIBis — ", 'OS-21-?'; riga 127, "OS21 classifica III Bis ", ?OS-21-III'; riga 302, "dunque in class. III-bis", viene assunta classifica III; altre: 305,372,374 riga 338, "IV-bis" assunto come IV; simili: 339,341,827;

Motivazioni:

1. la class_regex non accetta "IIIBis" come classifica;
2. la class_regex accetta solo la dicitura "IV bis".

Categorie non riconosciute dal dizionario:

```
riga 61, "OS 18", "OS18" e simili; altre istanze simili: 177,183,195
 riga 132, "OS  2"; altre: 134, 147,150,151,248
 riga 131, " OS 12, OS 18,"; altre: 133, 378,379,380,381
 riga 260, "OS12 anziché OS12B in quanto" , OS12 non accettato dal filtering; a
 riga 705, ["Os28" si chiede la ], viene estratto Os-28, la "s" minuscola non è
```

Motivazioni: Sono considerate valide le forme "OS-2A" e "OS-2B", "OS-12A" e "OS-12B", "OS-18A" e "OS-18B", "OS-20A" e "OS-20B"; assenti nel dizionario, invece, le categorie prive di specifica A/B ("OS-2", "OS-12", "OS-18", "OS-20").

Valori erronei di classifica economica

```
1. Numero romano estraneo
   riga 802, "OG1" per l'intero importo dell'appalto (e ovviamente per il Lotto
   riga 803, OS14" per il lotto II ;
   riga 804, OS22" per il lotto III o eventualmente ricorrere ad ATI o avvalimen
2. Altre maiuscole non relative a classifica
   riga 1100, "OS13 (S.I.O.S. scorporabile e subappaltabile max 30%)
3. Caratteri estranei
```

Pattern di scrittura molteplici I dati SOA, provenendo da un linguaggio burocratico, non sono scritti in maniera uguale in tutti i documenti ma soffrono di una forte variabilità. Vogliamo dare un'idea di quanti modi diversi esistono per indicare una categoria: per esempio OG-12, "Opere ed impianti di bonifica e protezione ambientale", può essere indicato in uno qualsiasi dei modi seguenti:

- Opere Generali 12;
- Opere Generali di tipo 12;
- Op.Gen. cat.12;
- og 12;
- og12;
- o.g. 12;
- og12;
- o.g.12;
- og.12;
- og. 12;

- og-12;
- o.g.-12;
- og.-12;
- OG 12, "Opere ed impianti di bonifica e protezione ambientale".

È altrettanto possibile che la categoria economica, normalmente espressa in numeri romani, si trovi scritta in modi differenti; vediamo il caso della III categoria, di seguito:

- categoria III, con importo fino a euro 1.033.000;
- cat.III;
- categoria terza;
- cat.3°.

Se si aggiunge che alcuni dei documenti sono stati digitalizzati a partire da fogli stampati e acquisiti con OCR, ci potrebbero essere degli errori di interpretazione; noi di seguito assumeremo di avere a disposizione un documento privo di errori di OCR.

5.7 Considerazioni

Gli errori rilevati nell'output - come detto precedentemente - sorgono in diversi punti della logica utilizzata e hanno livelli di problematicità differenti:

- un bug nello script: se sistematico, può essere corretto facilmente;
- un valore assente a livello di dizionario: questo comporta delle decisioni sui valori del dominio; se trovo la stringa "OS-18", è sensato supporre che si stia parlando di "OS-18A"? o forse è possibile che l'autore sottointendesse un valore "OS-18B" dopo averlo precedentemente menzionato?

A livello di regex è possibile apportare modifiche per fare match dei casi più particolari, però si pongono due problemi:

- dopo aver adattato la regex ad un nuovo caso particolare, questa potrebbe non accettare alcuni casi che facevano match in precedenza; dunque ad ogni modifica è necessario ricontrollare esaustivamente tutti gli output - compresi quelli che erano processati bene prima delle modifiche;

- eccessiva complessità della regex: dopo pochi rimaneggiamenti, l'espressione regolare diventa tendenzialmente incomprensibile, per cui future correzioni e modifiche diventano più difficili ed error-prone.

In generale i documenti esaminati usano un linguaggio naturale; è quindi possibile -nonostante la natura tecnica dei discorsi- che un documento da processare contenga attestazioni SOA trascritte in maniere differenti, non note a priori (Ad es. "OS11", "Opere specialistiche, categoria 11", "Op.Sp. 11"), dettate essenzialmente dallo stile dell'autore.

Dunque, in un documento potrebbe esserci una trascrizione nuova da adottare; e l'ideale sarebbe aggiungerla in maniera incrementale alla casistica di trascrizioni già adottate in precedenza. A livello di regex l'approccio incrementale non è per nulla agevole: si deve scrivere una regex nuova, che aggiunga i nuovi casi rilevati senza compromettere quelli precedenti. La nuova regex deve essere dunque testata sia sulla casistica di valori nuovi sia sulla casistica vecchia. Infine, rimane il problema che la regex non mi consente di delineare un contesto, ad es.: nella stringa "OS14 per il lotto II" la class_regex troverà un numero romano ("II", appunto) e lo assumerà erroneamente come valore di classifica. Risolvere questo singolo caso a livello regex -nello specifico, escludere i numeri romani dopo la parola "lotto"- comporta espressioni regolari più complesse e non esclude ulteriori casi di falsi-positivi.

5.8 Superare le limitazioni delle regex

In estrema sintesi, la criticità principale dell'approccio usato è affidare il riconoscimento del valore-stringa ad un criterio fissato a priori. L'approccio ideale dovrebbe consentire all'operatore umano di aggiungere istanze alla casistica già presente, in modo da:

- avere a disposizione l'interpretazione umana nei casi in cui l'automa riconoscitore della regex fraintenda un input;
- raccogliere tali istanze in maniera incrementale, in modo che il loro accumularsi costituisca una forma di esperienza applicabile automaticamente ai successivi input.

L'esperienza acquisita fondamentalmente permetterebbe di confrontare l'input non con un singolo pattern a priori -che nell'esempio specifico è una regex, ma potrebbe anche essere definito in maniera procedurale- ma con un insieme di più pattern, incrementabili a discrezione dell'operatore umano per catturare anche le istanze meno standardizzabili del linguaggio.

L'approccio del Machine Learning supervisionato consiste nell'usare un algoritmo per produrre un modello che minimizzi gli errori di classificazione rispetto a delle classificazioni vere per ipotesi. Questo ha una conseguenza importante: un

classificatore, una volta istruito a riconoscere una classe partendo da una certa quantità di esempi giusti, potrà stimare la classe anche in presenza di dati non uguali agli esempi di partenza. Ciò costituisce una grande differenza rispetto alla regex, che consente di rilevare un testo solo se esattamente conforme all'automa da essa descritto.

5.9 Annotazioni

Per poter applicare l'approccio ML alla ricerca di attestazioni SOA è necessario raccogliere le cosiddette **annotazioni gold** per costruire la **Ground Truth** sulla quale basare l'apprendimento della rete neurale.

Finora abbiamo introdotto le annotazioni **gold**. Esiste però anche la possibilità di adoperare altri tipi di annotazione, senza imporre che siano corrette per definizione; queste altre annotazioni, definite in letteratura come annotazioni **silver**[5], a differenza delle gold vanno usate con la consapevolezza che potrebbero descrivere delle entità non corrette; vanno usate dunque solo quando è possibile tollerare degli errori.

Per la realizzazione delle annotazioni è stato utilizzato l'apposito strumento di annotazione Doccano[7].

5.10 La libreria SpaCy

Per l'implementazione di una rete neurale abbiamo fatto uso della libreria **Spacy**, una libreria itopen-source scritta in Python e Cython che implementa funzionalità

di Natural Language Processing. Tra i progetti che appartengono al mondo SpaCy vi è la libreria Thinc, che permette di importare modelli statistici da PyTorch, TensorFlow e MXNet[2]. Spacy fornisce funzionalità di **tagger**, **parser**, **text categorizer**, **ner** e permette di articolarli in una pipeline; inoltre rende possibile la configurazione di nuovi componenti e la loro aggiunta alla pipeline.

5.11 Uso di Spacy

Per l'uso di SpaCy abbiamo dovuto creare gli insiemi di Training, Development e Test. Le annotazioni gold della ground truth sono state dunque divise in questi tre insiemi, rispettivamente aventi il 70%, il 10 % e il 20 % delle annotazioni golden.

Configurazione della pipeline: Istruisco SpaCy su quali sono le funzionalità da inserire in pipeline. Nel caso in questione la pipeline deve effettuare la tokenizzazione di elementi testuali della lingua italiana e la loro classificazione come entità. A livello di configurazione questo si traduce in queste righe del file base_config.cfg:

```
[nlp]
lang = "it"
pipeline = ["tok2vec","ner"]
```

Basta poi il seguente comando per ottenere la configurazione completa della pipeline:

```
spacy init fill-config base_config.cfg config.cfg
```

Fase di apprendimento: La costruzione del modello adopera il le golden labels di training e viene svolta col seguente comando:

```
spacy train config.cfg --paths.train ./train.spacy --paths.dev ./dev.
    spacy
  --output ./output
```

Debug: Data un'entità E, avere un numero di esempi di E troppo limitato significherebbe dare poca esperienza di E al modello.Per questo SpaCy fornisce un comando che controlla il numero di esempi per ogni entità, effettuando al contempo dei controlli ortografici su tutte le labels:

```
python3 -m spacy debug data config.cfg --paths.train ./train.spacy
  --paths.dev ./dev.spacy
```

Valutazione di precision e recall: Una volta istruito un modello ed esserci assicurati che le annotazioni che usa sono valide, possiamo mettere alla prova il modello: sottoponiamo al modello il docbin di test, le cui entità sono già annotate, e confrontiamo l'output del modello con le annotazioni gold. Tale confronto viene eseguito da Spacy con il seguente comando:

```
spacy evaluate output/model-best/ test.spacy --displacy-path . --
    displacy-limit 100
  --output output.json
```

Uso della rete neurale Dopo avere eseguito i passaggi precedenti che hanno permesso la configurazione e l'addestramento della NN il modello può essere adoperato per estrarre entità da documenti di testo.

Scorer Class Lo Scorer è la classe fornita da SpaCy per valutare le performance di un modello annotatore per ogni singola entità basandosi sulle annotazioni prodotte da tale modello. Lo Scorer richiede all'utente della libreria di avere a disposizione l'output del modello annotatore e le gold annotations della Ground Truth. Per ogni annotazione si crea un oggetto spacy.Span e si crea una lista di Span, che chiameremo labelled_entities_span_list; si potrà poi utilizzare la lista di gold annotations per creare il relativo **item** che descrive l'annotazione di un documento:

```
item = Example.from_dict(labelled_entities_span_list,
                         {"entities": [[start_offset, end_offset, word]
                                       for [start_offset, end_offset,
    word]
                                       in gold_annots]})
```

Listing 5.1: Esempio di item

Si crea un Example per ogni documento annotato e si pongono gli item in una **item_list**. Tutti i dati annotati dal modello annotatore sono ora nella item list ed è dunque possibile invocare il task di score vero e proprio:

```
    scores = scorer.score(item_list)
```

Capitolo 6

5. Valutazione dei risultati

Dopo l'implementazione dei prototipi NER_regex e NER_ML, possiamo introdurre la loro valutazione con l'uso di un apposito Scorer. Dato un sistema NER, lo Scorer fornisce delle misure di affidabilità dell'NER stesso in termini di **precision**, **recall** e **accuracy**.

6.1 Confronto tra regex e ML

Nella tabella scores-comparisons.pdf 6.1 riportiamo i valori di Precision, Recall e Accuracy ottenuti per ogni entità contemplata nel task di classificazione, sia nel caso di NER_regex che nel caso NER_ML. Le regex considerate sono le seguenti:

$$regex - soa_{wl-v1} = 'O(S|G)(dd?)'$$

$$regex - class_{wl-v1} = '(IV|IV|III|II|IX|VIII|VII|VI|X|V|I)'$$

Si può facilmente notare come molte entità di tipo **Categoria** vedono valori di precision recall e accuracy migliori nell'approccio Machine Learning; in particolare sono stati evidenziati in verde i valori di precision, recall ed f1-score che in NER_ML superano almeno dello 0.05 i corrispondenti valori registrati dall'NER_regex. In un caso (entity type OG-4) si registra un peggioramento delle performance del Machine Learning, che abbatte a zero le score. Questo può essere spiegato con la mancanza di esempi numerosi per alcune entità: ciò porterebbe il modello a funzionare molto bene nel riconoscere le entità ben rappresentate dalla ground truth, ma anche a funzionare peggio per tutte le entità meno note.

ENTITY TYPE	REGEX NER	P	R	F		ML NER	P	R	F
OG-1		1,00	0,71	0,83			0,99	0,98	0,98
OG-2		1,00	0,61	0,76			1,00	0,97	0,99
OG-3		1,00	0,43	0,60			0,98	0,90	0,94
OG-4		1,00	0,50	0,67			0,00	0,00	0,00
OG-5		0,00	0,00	0,00			0,00	0,00	0,00
OG-6		1,00	0,33	0,50			0,90	1,00	0,95
OG-7		0,00	0,00	0,00			0,00	0,00	0,00
OG-8		1,00	0,17	0,29			1,00	1,00	1,00
OG-9		1,00	0,94	0,97			1,00	0,94	0,97
OG-10		1,00	0,23	0,38			1,00	1,00	1,00
OG-11		0,99	0,65	0,79			1,00	0,97	0,98
OG-12		1,00	0,75	0,86			1,00	0,95	0,97
OG-13		0,00	0,00	0,00			1,00	0,67	0,80
OS-1		1,00	0,79	0,88			1,00	0,86	0,92
OS-2A		1,00	0,29	0,45			0,89	0,67	0,76
OS-2B		0,00	0,00	0,00			1,00	1,00	1,00
OS-3		1,00	0,62	0,77			1,00	0,99	0,99
OS-4		1,00	0,40	0,57			0,96	0,88	0,92
OS-5		1,00	0,50	0,67			1,00	1,00	1,00
OS-6		0,99	0,76	0,86			0,99	0,98	0,98
OS-7		1,00	0,66	0,79			0,93	0,81	0,86
OS-8		0,92	0,50	0,65			1,00	0,73	0,84
OS-9		0,00	0,00	0,00			0,50	1,00	0,67
OS-10		1,00	0,24	0,38			1,00	1,00	1,00
OS-11		1,00	0,17	0,29			1,00	1,00	1,00
OS-12A		0,33	0,06	0,10			0,85	0,65	0,73
OS-12B		0,00	0,00	0,00			1,00	1,00	1,00
OS-13		1,00	0,50	0,67			0,94	1,00	0,97
OS-14		1,00	0,68	0,81			1,00	1,00	1,00

Figura 6.1: confronto tra l'approccio regex(sinistra) e NN(destra)

Come sottolineato precedentemente, l'approccio NER_regex non ha grandi margini di miglioramento dell'accuratezza: in sintesi è un approccio rigido, che in caso di inclusione di nuovi pattern testuali richiederebbe la continua modifica di una regex in forme sempre più complesse, risolvendosi in problemi di manutenzione, di testing, di leggibilità e talvolta di sicurezza. Nel caso del Machine Learning un task di Named Entity Recognition (NER) può avere risultati migliori delle regex, anche se può inizialmente soffrire di scarsa numerosità di entities. Su questa base si potrebbe proporre l'uso combinato degli output di entrambi i modelli: in particolare, si potrebbe ricorrere al modello regex per le entità meno assimilate dal modello Machine Learning (ML) , preferendo invece quest'ultimo per le entità su cui gli score descrivono prestazioni migliori.

Nell'istogramma in figura 6.2 si confrontano le medie di P,R,F1 nel caso della regex e nel caso dell'NER_ML. Le regex fin qui considerate sono regex non complicate, possono certamente essere migliorate per coprire una casistica di scritture più estesa: prevedendo degli spazi aggiuntivi, prevedendo separatori alternativi e prevedendo l'uso combinato di lettere maiuscole e minuscole. Tuttavia, come sottolineato precedentemente, caratteristica fondamentale dei dati è essere usati nella letteratura burocratica: in quest'ambito, colui che scrive il documento formale può

ad esempio scrivere “OG4”, ma anche “Opera Generale numero quattro” e ancora “Op.Gen.num.4”, per cui risulta ingiustificato investire tempo per ottenere una regex “ottimale”. Un’alternativa potrebbe nascere dalla combinazione dell’uso del NER_regex e del NER_ML.

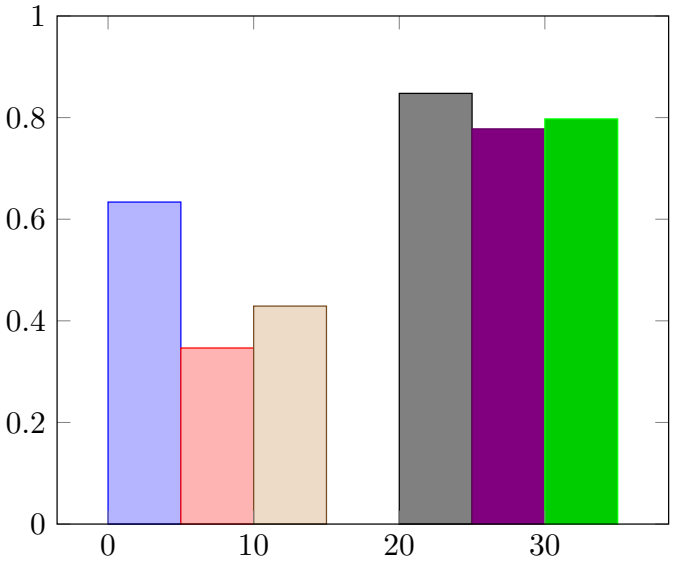

Figura 6.2: Da sinistra: Valori di P,R,F del NER_regex-v1 e del NER_ML

6.2 Human In The Loop

Fin qui il lavoro di SOA extraction ci ha portati a considerare due tecnologie molto diverse per approccio e manutenibilità.
In particolare, le **regex** risultano di risultato immediato poiché in pochi minuti ci consentono di svolgere un’estrazione di dati SOA con discreti valori di recall, ma soffrono di **scarsa leggibilità**, manutenzione problematica e risultano perciò decisamente **error-prone**; nonostante un incoraggiante risultato immediato, non permettono di migliorare in maniera efficiente l’estrazione di entità per tutta una serie di fattori. Consideriamo ad esempio il manutentore a cui toccherà estendere la regex in produzione reg_prod_n: il suo compito sarà tipicamente quello di trasformare la reg_prod_n in una reg_prod_{n+1} in maniera da includere i nuovi pattern $p_{m+1}, \ldots, p_{m+k}$. Le domande che ci poniamo sono allora le seguenti:

- Quella vecchia regex reg_prod_n sarà stata fino ad allora documentata per tenere traccia dei pattern precedenti $p_1, \ldots, p_m$?
- Oppure spetterà al programmatore ricordare tutti i singoli pattern che venivano catturati con la precedente versione della regex?
- E la regex avrà dei dati di **testing**, per verificare che la nuova versione reg_prod_{n+1} non comprometta i pattern fino ad allora accettati?

Tutte queste domande si ritrovano nell’indagine statistica di “Regexes are hard” [6], dove emerge che le best practices dell’uso delle regex sono sistematicamente

disattese: le regex vengono scritte perlopiù senza aggiungere documentazione e senza dati di test. Tutte queste problematiche nel complesso impediscono ad un sistema basato su regex di essere efficiente nel miglioramento della Precision.
Al contrario, le reti neurali possono nel tempo "imparare meglio" le entità con cui hanno a che fare ma richiedono un consistente lavoro umano di annotazione e classificazione delle entità presenti nei dati stessi; in altre parole, mancano dell'immediatezza delle regex ma rendono fattibile il **fine-tuning** del sistema classificatore.
I rispettivi punti di forza e di debolezza delle due metodologie risultano complementari, per cui ha senso combinarli in un unico sistema che sfrutti i vantaggi di entrambe, mitigandone gli svantaggi.

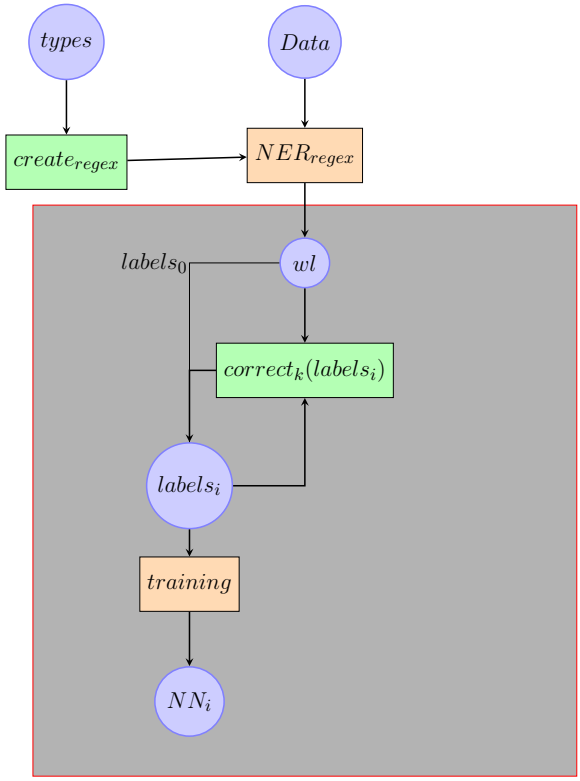

Figura 6.3: Framework HITL: fase di addestramento

Configuriamo l'uso delle componenti regex, ML, task di creazione regex e task di annotazione manuale come un sistema di tipo HITL. In tale sistema l'essere umano può essere utilizzato come programmatore di regex e come annotatore a più riprese successive. Partendo dallo studio "How to invest my time" [10], l'indicazione che traiamo per usare al meglio il lavoro umano è investire pochi minuti sulla creazione di regex ad alta Recall per poi effettuare la correzione umana delle annotazioni prodotte dalla regex stessa. Seguendo la convenzione dello studio citato, chiameremo

"weak labelling" l'estrazione svolta con la regex, che indicheremo con RE_{WL}: le label così prodotte risultano "weak", ossia non verificate e passibili d'errore. L'attività umana $correct_k labels$, partendo dalle weak labels come input iniziale $labels_0$, ad ogni iterazione i apporterà k correzioni che andranno a formare l'insieme di labels $labels_i$. Qualora il ciclo continui, tale insieme i-esimo potrà essere adoperato come input dell'iterazione $i+1$; in caso contrario verrà assunto come training set per istruire la rete neurale.

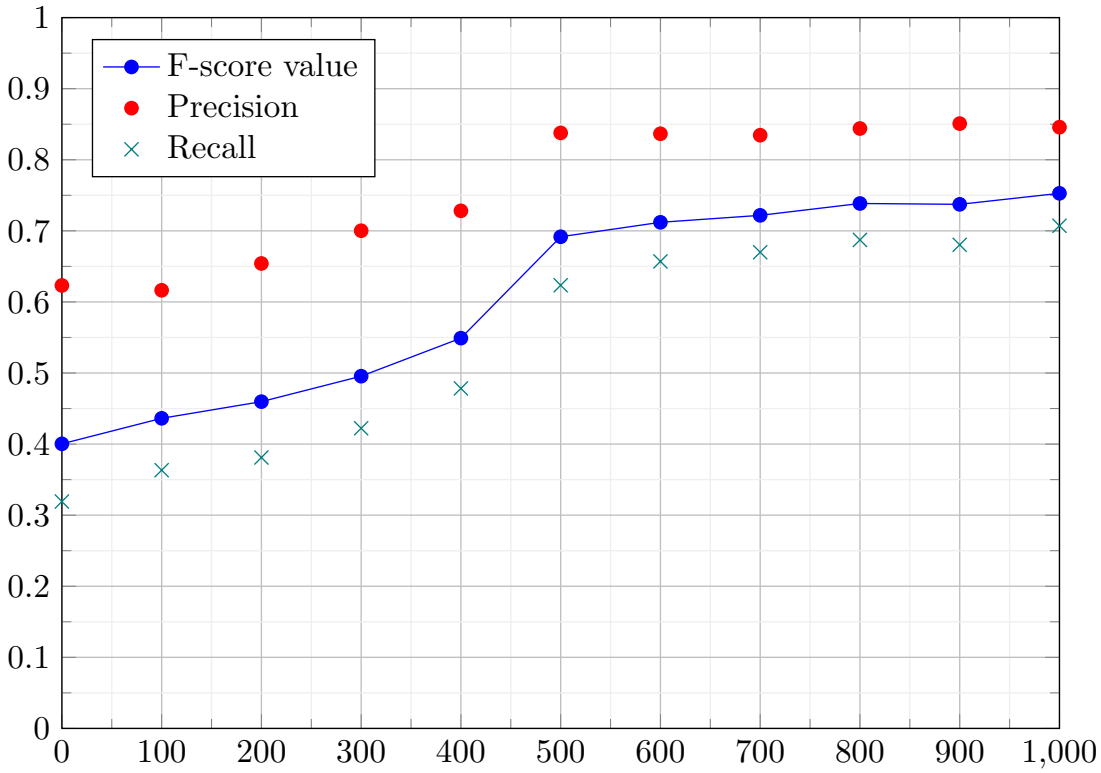

Figura 6.4: Human In The Loop: F1-score, Precision e Recall.

Il risultato del loop è l'addestramento della rete neurale n-esima NN_n utilizzabile per effettuare il task di NER. Chiameremo NER_HITL_n il task di NER condotto con la NN_n così prodotta a valle del loop.
L'uso dello Human In The Loop può essere vantaggioso se effettuato in maniera tale da:

- istruire una rete neurale in un tempo minimo di lavoro umano;
- permettere il fine-tuning della rete neurale stessa con l'intervento dell'annotatore umano.

Seguendo dunque lo studio già citato, l'operatore umano deve destinare pochi minuti alla realizzazione delle regex RE_{WL}. Nella nostra prova utilizziamo le regex

$$regex - soa_{wl-v1} = \text{'}O(S|G)(dd?)\text{'}$$

$$regex - class_{wl-v1} = \text{'}(IV|IV|III|II|IX|VIII|VII|VI|X|V|I)\text{'}$$

realizzate in pochi minuti su un tool di regex-editing liberamente disponibile online. Usando tali regex abbiamo quindi prodotto le weak labels che vanno a formare

il dataset iniziale $labels_0$; per ogni iterazione successiva $i > 0$ vengono dunque corrette k annotazioni weak con $k = 100$. Il grafico 6.4 illustra l'evoluzione del sistema HITL all'aumentare delle correzioni umane apportate alle weak labels: viene rappresentato il guadagno in termini di Precision, Recall e F1-score. In vari esperimenti analoghi a quello riportato in figura 6.4 si evince come in dieci iterazioni di correzione il sistema valutato in termini di P, R, F1 riporti i guadagni minimi:

$$\Delta(p)_{i=10} = (22\%)$$

$$\Delta(r)_{i=10} = (34\%)$$

$$\Delta(f)_{i=10} = (32\%)$$

annot.$_{gold}$	p	r	f	p - p$_{ML}$	r - r$_{ML}$	f - f$_{ML}$
0	0.6231	0.3191	0.4003	-0.2244	-0.4586	-0.3970
100	0.6163	0.3632	0.4362	-0.2312	-0.4145	-0.3611
200	0.6541	0.3811	0.4597	-0.1935	-0.3966	-0.3376
300	0.7001	0.4222	0.4955	-0.1475	-0.3555	-0.3018
400	0.7281	0.4783	0.5490	-0.1195	-0.2995	-0.2483
500	0.8377	0.6234	0.6917	-0.0098	-0.1543	-0.1056
600	0.8365	0.6569	0.7120	-0.0110	-0.1208	-0.0853
700	0.8345	0.6698	0.7218	-0.0130	-0.1079	-0.0755
800	0.8439	0.6872	0.7385	-0.0036	-0.0905	-0.0588
900	0.8508	0.6803	0.7373	0.0032	-0.0974	-0.0600
1000	0.8458	0.7071	0.7527	-0.0018	-0.0706	-0.0446

Figura 6.5: I valori di p,r,e f1 di NER_HITL-v1 e scarti rispetto a NER_ML

È da notare come l'iterazione numero dieci registri i valori:

$$P_i = 0.8458$$

$$R_i = 0.7071$$

$$F_i = 0.7527$$

a fronte dei valori del sistema NER_ML costruito sulla *ground truth*

$$P_{NER\_ML} = 0.8476$$

$$R_{NER\_ML} = 0.7778$$

$$F_{NER\_ML} = 0.7973$$

Possiamo osservare lo stesso specifico esperimento alla tabella 6.5, dove forniamo anche un calcolo degli scarti tra i valori del NER_ML e i valori dello HITL ad ogni iterazione i. Si nota come le prestazioni di NER_HITL_n tendano a migliorare all'aumentare delle iterazioni n.

I dataset adoperati: Nel caso ML sono state necessarie $annot - ML = 2218$ annotazioni gold prodotte da un annotatore umano; di questo numero di annotazioni totale, 1500 vanno a costituire il training set ML. Considerando invece il caso del sistema HITL, vengono corrette $annot - HITL_{10} = 1428$ annotazioni silver, di cui 1000 sono impiegate al fine di correggere le annotazioni precedentemente prodotte con regex (1315 annotazioni prodotte con regex). Dunque l'esperimento HITL adopera internamente $annot - ML(annot - HITL_{10} = 790$ annotazioni in meno. Va considerato però il tempo di produzione delle annotazioni gold e $silver_{corrected}$: se la produzione di un'annotazione gold richiede $t_{gold-single-annot} = 14s$, la correzione di un'annotazione silver richiede soli $t_{single-silver-correct} = 6{,}52$ secondi mediamente. Confrontando dunque il tempo usato dall'operatore umani per i due scenar di annotazione, risulta che abbiamo risparmiato $t_{ML-annot} - t_{silver-correct} = (2218 \cdot 14)s - (1300 \cdot 6.52)s = 22576s$ pari a più di 6 ore per la produzione del dataset. In questo senso, ricorrere al sistema HITL fino all'iterazione i=10 fa risparmiare una percentuale pari al 72% del tempo totale di annotazione.

HITL-14: Se la precision p_{10} del sistema NER_HITL_{10} è soddisfacente, abbiamo mostrato una r_{10} che dello 0.70, pari a 7 punti percentuali inferiori rispetto alla NER_ML. Possiamo dunque decidere di incrementare il numero di correzioni a dataset e proseguire con ulteriori iterazioni. All'iterazione $i = 14$ abbiamo ottenuto il superamento delle prestazioni misurate dalla NER_ML. Si registrano i valori:

$$p_{H14} = 0.8823, r_{H14} = 0.7723, f_{H14} = 0.8141$$

Per raggiungere l'iterazione 14 è stato necessario fornire $annot - HITL_{10} = 2000s$ annotazioni $silver_{corrected}$. La loro produzione richiede un tempo pari a $t_{silver-correct} = 2000 \cdot 6.52s$, che confrontato a $t_{ML-annot}$ comporta un risparmio del 58% del tempo di annotazione del NER_ML.
Possiamo constatare che nel complesso lo schema HITL ha permesso di realizzare due esperimenti alquanto interessanti. Se la resa di HITL-10 risulta infatti non ottimale, dobbiamo tenere in considerazione che apporta il vantaggio di ridurre drasticamente il tempo umano d'annotazione (72% in meno rispetto a $time_{annotML}$ avvalendosi di un numero di annotazioni $n_{annotHITL10} = 1300$ molto inferiore al numero di annotazioni $n_{annotML} = 2218$ utilizzate nell'approccio NER_ML puro. È da notare l'iterazione HITL-14 che, adoperando un numero di annotazioni pari a$n_{annotHITL14} = 2000$ maggiore rispetto al caso $i = 10$, ottiene delle performance superiori al NER_ML richiedendo un tempo d'annotazione molto minore rispetto al NER_ML stesso.

Capitolo 7

Conclusioni

Scopo di questa tesi è stato effettuare un task di Named Entity Recognition su documenti provenienti dalla pubblica amministrazione; tale set di documenti, rumorosi in un quanto spesso ottenuti via OCR a partire da documenti stampati, risulta spesso descrivere gli stessi dati formali con diciture variabili. È stata presentata l'implementazione del task di NER con l'uso di regex, ma è apparso chiaro come questo approccio risulti difficile da migliorare nel tempo; infatti le regex alla modifica risultano error-prone e di difficile manutenzione e documentazione, caratteristiche che non aiutano ad esprimere pattern molto variabili. D'altro canto è stato presentato un NER che adopera tecniche di supervised deep learning e che meglio si adegua ad estrarre dati variabili, a spese tuttavia di un ingente tempo umano di annotazione. È apparsa dunque conveniente la possibilità di effettuare una "weak supervision" adoperando un dataset prodotto dal task di NER_regex; tale dataset viene prodotto in pochi minuti, dopodiché è sottoposto a cicliche iterazioni di correzione da parte di un annotatore umano. Il loop di addestramento così configurato è detto Human In The Loop. Abbiamo notato come l'operatore umano sia significativamente più veloce nel compito di *correzione-annotazione* rispetto al compito di produzione di annotazioni *from scratch*. Valutando le perfomance di NER_HITL_i al variare dell'iterazione i-esima, è emerso che: è possibile ottenere un risultato subottimale con la sola *annotazione-correzione* di 1000 annotazioni, sacrificando 7% di recall ma comunque risparmiando più del 70% del tempo d'annotazione. Infine, portando le iterazioni di NER_HITL_i a 14 è possibile superare le prestazioni di NER_ML, adoperando comunque il 10% di annotazioni in meno e risparmiando un tempo di annotazione pari a 58%. È possibile immaginare dei futuri miglioramenti per lo schema Human In The Loop individuato; in particolare si potrebbe fornire all'annotatore una metrica per calcolare quali porzioni di testo sono più rumorose e probabilmente più prioritarie rispetto al task di annotazione-correzione .

Acronimi

A Accuracy. 13

CIG Codice Identificativo di Gara. 6

CUP Codice Unico di Progetto. 6

F1 F1-score. 35

FN False Negative. 13

FP False Positive. 12, 13

HITL Human In The Loop. 15, 17, 33–35

I classifica fino a euro 258.000. 9

II classifica fino a euro 516.000. 9

III bis classifica fino a euro 1.500.000. 9

III classifica fino a euro 1.033.000. 9

IV bis classifica fino a euro 3.500.000. 9

IV classifica fino a euro 2.582.000. 9

ML Machine Learning. 15, 31, 33

NER Named Entity Recognition. 15, 17, 31, 34

NN Neural Network. 15, 16, 29

OG-1 Edifici civili e industriali. 7

OG-10 Impianti per la trasformazione alta/media tensione e per la distribuzione di energia elettrica in corrente alternata e continua ed impianti di pubblica illuminazione. 7

OG-11 Impianti tecnologici. 7

OG-12 Opere ed impianti di bonifica e protezione ambientale. 7

OG-13 Opere di ingegneria naturalistica. 7

OG-2 Restauro e manutenzione dei beni immobili sottoposti a tutela. 7

OG-3 Strade, autostrade, ponti, viadotti, ferrovie, metropolitane. 7

OG-4 Opere d'arte nel sottosuolo. 7

OG-5 Dighe. 7

OG-6 Acquedotti, gasdotti, oleodotti, opere di irrigazione e di evacuazione. 7

OG-7 Opere marittime e lavori di dragaggio. 7

OG-8 Opere fluviali, di difesa, di sistemazione idraulica e di bonifica. 7

OG-9 Impianti per la produzione di energia elettrica. 7

OS-1 Lavori in terra. 7

OS-10 Segnaletica stradale non luminosa. 8

OS-11 Apparecchiature strutturali speciali. 8

OS-12-A Barriere stradali di sicurezza. 8

OS-12-B Barriere paramassi, fermaneve e simili. 8

OS-13 Strutture prefabbricate in cemento armato. 8

OS-14 Impianti di smaltimento e recupero rifiuti. 8

OS-15 Pulizia di acque marine, lacustri, fluviali. 8

OS-16 Impianti per centrali produzione energia elettrica. 8

OS-17 Linee telefoniche ed impianti di telefonia. 8

OS-18-A Componenti strutturali in acciaio. 8

OS-18-B Componenti per facciate continue. 8

OS-19 Impianti di reti di telecomunicazione e di trasmissioni e trattamento. 8

OS-2-A Superfici decorate di beni immobili del patrimonio culturale e beni culturali mobili di interesse storico, artistico, archeologico ed etnoantropologico. 7

OS-2-B Beni culturali mobili di interesse archivistico e librario. 7

OS-20-A Rilevamenti topografici. 8

OS-20-B Indagini geognostiche. 8

OS-21 Opere strutturali speciali. 8

OS-22 Impianti di potabilizzazione e depurazione. 8

OS-23 Demolizione di opere. 8

OS-24 Verde e arredo urbano. 8

OS-25 Scavi archeologici. 8

OS-26 Pavimentazioni e sovrastrutture speciali. 8

OS-27 Impianti per la trazione elettrica. 8

OS-28 Impianti termici e di condizionamento. 8

OS-29 Armamento ferroviario. 8

OS-3 Impianti idrico-sanitario, cucine, lavanderie. 7

OS-30 Impianti interni elettrici, telefonici, radiotelefonici e televisivi. 8

OS-31 Impianti per la mobilitÃ sospesa. 8

OS-32 Strutture in legno. 8

OS-33 Coperture speciali. 8

OS-34 Sistemi antirumore per infrastrutture di mobilitÃ . 8

OS-35 Interventi a basso impatto ambientale. 8

OS-4 Impianti elettromeccanici trasportatori. 7

OS-5 Impianti pneumatici e antintrusione. 7

OS-6 Finiture di opere generali in materiali lignei, plastici, metallici e vetrosi. 7

OS-7 Finiture di opere generali di natura edile e tecnica. 7

OS-8 Opere di impermeabilizzazione. 7

OS-9 Impianti per la segnaletica luminosa e la sicurezza del traffico. 8

P Precision. 13, 33, 35

PA Pubblica Amministrazione. 4, 5, 10

R Recall. 13, 33, 35

SOA Società Organismi di Attestazione. 6, 20, 32

TN True Negative. 12, 13

TP True Positive. 12, 13

V classifica fino a euro 5.165.000. 9

VI classifica fino a euro 10.329.000. 9

VII classifica fino a euro 15.494.000. 9

VIII classifica oltre euro 15.494.000. 9

Bibliografia

[1] A. Basacchi. *Codice civile. Il nuovo codice civile aggiornato*. Kollesis Editrice, 2013.

[2] ExplosionAI GmbH. https://thinc.ai/docs/usage-frameworks.

[3] gazzettaufficiale.it. https://www.gazzettaufficiale.it/atto/serie_generale/caricaDettaglioAtto/originario?atto.dataPubblicazioneGazzetta=2000-02-29&atto.codiceRedazionale=000G0071&elenco30giorni=false.

[4] IQAndreas. https://softwareengineering.stackexchange.com/questions/223634/what-is-meant-by-now-you-have-two-problems.

[5] Pierre André Ménard and Antoine Mougeot. Turning silver into gold: error-focused corpus reannotation with active learning. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 758–767, Varna, Bulgaria, September 2019. INCOMA Ltd.

[6] Louis G. Michael, James Donohue, James C. Davis, Dongyoon Lee, and Francisco Servant. Regexes are hard: Decision-making, difficulties, and risks in programming regular expressions. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 415–426, 2019.

[7] Hiroki Nakayama, Takahiro Kubo, Junya Kamura, Yasufumi Taniguchi, and Xu Liang. doccano: Text annotation tool for human, 2018. Software available from https://github.com/doccano/doccano.

[8] T. L. Packer and D. W. Embley. Cost effective ontology population with data from lists in ocred historical documents. In *Proceedings of the 2nd International Workshop on Historical Document Imaging and Processing*, pages 44–52, 2013.

[9] T. L. C. D. Silva, Regis Pires Magalhães, J. Macêdo, David Araújo, Natanael Araújo, Vinicius de Melo, Pedro Olímpio, P. Rego, and A. V. L. Neto. Improving named entity recognition using deep learning with human in the loop. In *EDBT*, 2019.

[10] Shanshan Zhang, Lihong He, Eduard Dragut, and Slobodan Vucetic. How to invest my time: Lessons from human-in-the-loop entity extraction. KDD '19, pages 2305–2313, New York, NY, USA, 2019. Association for Computing Machinery.