Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

SUGGERITORE DI RIFERIMENTI NORMATIVI NELLA LEGISLAZIONE EUROPEA: IL CASO DI STUDIO DI LEOS

Relatore:

Chiar.mo Prof.

Vitali Fabio

Correlatore:

Chiar.ma Prof.ssa

Monica Palmirani

Presentata da:

Emanuele Di Sante

III Sessione

Anno Accademico 2022/2023

Legal XML,
NLP,
Similarità tematica

Dedicato a Luciano e Maria Teresa,
ai miei amici e compagni di corso,
ai colleghi del team Leos

Abstract

Il progetto di tesi mira a ottimizzare il processo legislativo europeo tramite l'utilizzo delle nuove tecnologie dell'informazione, focalizzandosi sullo sviluppo di un componente server-side per assistere i funzionari della Commissione Europea nella redazione dei riferimenti normativi. Questo lavoro, condotto dal centro di ricerca CIRSFID-ALMA-AI dell'Università di Bologna, punta a migliorare la qualità e l'efficienza della stesura di documenti legislativi, fornendo strumenti di supporto per gli esperti di legal drafting della Commissione Europea. Il NormRefAI assistant, componente per i riferimenti normativi, mira a garantire la consistenza e la correttezza dei riferimenti, facilitando il lavoro dei funzionari e contribuendo a una maggiore precisione e affidabilità nel processo decisionale e nella produzione dei testi legislativi. La valutazione del NormRefAI assistant è stata condotta congiuntamente a giuristi esperti del CIRSFID-ALMA-AI, e i risultati dei test sono stati validati da due giuristi. Sviluppi futuri includono l'integrazione di nuove funzionalità e il miglioramento dei controlli di validità dei suggerimenti.

Indice

Capitolo 1 - Introduzione

Lo scopo di questa dissertazione è illustrare come il processo legislativo europeo possa beneficiare dalle nuove tecnologie dell'informazione per agevolare la scrittura di nuove leggi ed evitare errori. A tal proposito è stato realizzato un componente server-side che mira ad assistere i funzionari della Commissione Europea nella redazione di riferimenti normativi. La ricerca è portata avanti dal centro di ricerca CIRSFID-ALMA-AI dell'università di Bologna, ed ha come obiettivo ultimo quello di migliorare la qualità e l'efficienza della stesura di documenti legislativi. Il DISI è parte integrante di questo progetto mediante il prof. Fabio Vitali responsabile di integrare i servizi server-side in una interfaccia client-side di LEOS secondo i principi di usabilità.
Nel contesto del processo legislativo europeo, emerge una sfida durante le prime fasi, ossia cercare di evitare sovrapposizioni o conflitti con la normativa europea già per questo il metodo delle citazioni normative, ossia richiamare altre norme già esistenti nel sistema legislativo, è una tecnica di legal drafting consolidata da secoli nelle discipline giuridiche.
Gli errori nel creare riferimenti normativi sono molteplici e possiamo classificarli come segue:

1. Riferimenti a legislazione non più in vigore ossia che è stata abrogata (e.g., direttiva abrogata);
2. Riferimenti a legislazione che ha modificato la struttura e quindi le partizioni sono state rinumerate (e.g., art. 4 che è diventato art. 7);
3. Riferimenti che puntano a un documento ma questo è stato totalmente ripubblicato e riformato (e.g., regolamento che è stato abrogato totalmente e sostituito con un altro- fenomeno del recasting);
4. Riferimenti che a causa delle riforme delle istituzioni europee sono sintatticamente scritti diversamente (e.g., Regulation No 1009/67/EEC, Regulation (EU) 2016/679);
5. riferimenti simili ma di materie completamente diverse (Regulation (EC) No 1223/2009, Regulation (EC) No 223/2009).

I protagonisti principali di questo processo sono i funzionari della Commissione Europea nella prima stesura. Intervengono successivamente altri attori quali il Parlamento e il Consiglio dell'Unione europea. Esistono in Commissione Europea più di

500 uffici dedicati alla redazione delle proposte legislative che sono composte da professionisti in settori specifici (e.g., agricoltura, tasse, economia). Nonostante la loro conoscenza della legislazione sia approfondita, il processo implica molti passaggi e questo è un ulteriore elemento che incrementa la possibilità di commettere errori. Il problema dei riferimenti normativi errati ha un impatto notevole nella parte istruttoria della legislazione definita Analisi dell'Impatto della Regolazione cha ha lo scopo di comprendere se vi sono contrasti normativi in essere nel corpus delle leggi europee o internazionali. Inoltre in caso di atti modificativi un errato riferimento comporta a catena errori di consolidamento e modifiche riportate male. Questo studio si propone di fornire strumenti di supporto per migliorare il processo di scrittura legislativa dei riferimenti normativi orientato agli esperti di legal drafting della Commissione Europea, in particolare coloro che lavorano dentro alle Direzioni Generali[1]. Il presente lavoro è stato realizzato all'interno di un progetto in collaborazione con la DG Informatics della Commissione EU.

Ricercando il contesto, si esplora uno studio condotto dall'Università di Bologna incentrato sull'utilizzo delle tecnologie dell'informazione per ottimizzare il processo legislativo. Tale studio, nominato "Legal Drafting in the Era of Artificial Intelligence and Digitisation", mira a sviluppare un framework legislativo basato sui principi della filosofia del diritto, dell'informatica giuridica, dell'intelligenza artificiale e della linguistica computazionale, al fine di rendere il diritto computabile e legalmente autorevole. L'approccio adottato integra sia applicazioni di intelligenza artificiale simboliche che non simboliche, utilizzando Akoma Ntoso come standard LegalXML per fornire una base comune per le applicazioni AI robuste. Akoma Ntoso segue il principio di "auto-contenimento", garantendo che tutti i metadati e le conoscenze siano rappresentati nella stessa struttura logica XML, facilitando l'accesso alle informazioni richieste accedendo solamente al documento XML. Questo principio agevola l'applicazione di tecniche di AI che possono quindi sfruttare testo, annotazioni, metadata "in-place" nel documento.

L'obiettivo principale dell'implementazione di questo strumento è di assistere i funzionari della Commissione europea nel loro compito di legal drafting delle proposte legislative (regolamenti, direttive, decisioni) che poi andranno in discussione presso il Parlamento europeo e il Consiglio dell'Unione Europea, professionisti del settore che, pur non agendo direttamente come legislatori, trarranno vantaggio dall'utilizzo del componente sviluppato in questa tesi. Questo strumento mira a fornire loro un valido

[1] La Commissione Europea è suddivisa organizzativamente in Divisioni Generali, ciascuna con compiti e ambiti ben precisi. https://publications.europa.eu/code/it/it-390600.htm

supporto nella redazione dei documenti, confermando, segnalando o integrando i riferimenti normativi indicati. Ciò migliorerà significativamente la loro esperienza utente, garantendo una maggiore precisione e affidabilità nel processo decisionale e nella produzione dei testi legislativi.
Il componente di suggerimento dei riferimenti normativi (da ora denominato NormRefAI assistant)si propone di fornire una consistenza nella forma della stesura dei riferimenti, analizzare la correttezza dei riferimenti completi, suggerire correzioni per i riferimenti incompleti o poco coerenti e fornire informazioni utili per verificare l'eventuale correttezza rispetto alla tematica del documento.
Si devono quindi prendere in considerazione tre caratteristiche principali:

1. sintassi (completa, incompleta, errata secondo il periodo storico);
2. la validità giuridica del riferimento nel tempo;
3. la classificazione semantica e tematica del documento..

La valutazione è stata svolta insieme a giuristi esperti appartenenti al gruppo di lavoro del CIRSFID-ALMA-AI e condotti sotto la supervisione della prof.ssa Monica Palmirani. La valutazione ha fornito dati interessanti che hanno attestato la consistenza e affidabilità delle risposte dell'applicativo.
Si è scelto un insieme di riferimenti significativi e si sono testati rispetto alla consistenza rispetto a quelli suggeriti dal NormRefAI assistant.
Nel NormRefAI assistant, l'utilizzo di strumenti AI affiancati da meccanismi di validazione dei risultati è particolarmente interessante, garantendo risposte significative e prive di errori che sicuramente forniranno un valido supporto all'utente finale.
Nonostante la funzionalità già buona del componente, ci sono margini di miglioramento. Integrare nuove funzionalità basate su una LLM in fase di sviluppo permetterà di sperimentare nuove caratteristiche come l'autocomposizione del riferimento normativo usando il contesto. Inoltre, sono previsti controlli di validità aggiuntivi per i suggerimenti, inclusa la verifica tramite qualificatori e data di validità che per ora per mancanza di documenti nel dataset non sono ancora stati implementati. Aggiungere la possibilità di segnalare errori e inconsistenze nei riferimenti, nonché identificare il formato corretto di citazione in base alla data di rilascio del documento, contribuirà a migliorare ulteriormente il componente.

Capitolo 2 - Tecnologie per il processo legislativo

2.1 - Origine e motivi della ricerca

Il NormRefAI assistant è parte di uno studio svolto dall'università di Bologna, CIRSFID-ALMA-AI che esplora le potenzialità delle tecnologie dell'informazione per migliorare la qualità e l'efficienza del processo legislativo. Il DISI è parte integrante di questo progetto mediante il prof. Fabio Vitali responsabile di integrare i servizi server-side in una interfaccia client-side di LEOS secondo i principi di usabilità.
Negli ultimi venti anni il processo di digitalizzazione delle risorse legali, in particolare quelle legislative, ha subito molte evoluzioni. Allo stesso modo la ricerca da parte delle comunità AI e del diritto si è fortemente evoluta nel tempo. Sono state sviluppate teorie e metodi per modellare norme in formule giuridiche e per gestire l'interazione del ragionamento legale con esperti giuridici tra cui Guido Governatori, Monica Palmirani, Guido Boella, Leos Van der Torre, Tom Van Engers [PV22].
In particolare, è stata esplorata una strada riconosciuta come "Law as Code", che mira a migliorare la ricerca dell'informazione da un qualsivoglia testo legale tramite l'uso di ontologie giuridiche e semantiche (Palmirani 2018, 2019, 2020). Tale movimento è stato incluso nell'agenda dell'interoperabilità della Commissione europea[2].
Possiamo citare molti progetti in cui questo concetto si realizza:

- "The Lynx project" (https://lynx-project.eu/), che si pone l'obbiettivo di tradurre un sistema giuridico in un knowledge graph, ma non mantiene il collegamento con le risorse giuridiche testuali ufficiali quali le Gazzette;
- "ManyLaws" (https://www.manylaws.eu/), ricerca che mira a codificare la normativa in linguaggi di programmazione senza passare attraverso linguaggi giuridici, ma che comunque ha il limite di dover in ogni caso sincronizzare documenti eterogenei provenienti da diversi paesi in un portale ricercabile secondo principi giuridici diversi;

[2] 'Law as code' and interoperability | Joinup

- "OpenFisca" (https://openfisca.org/en/), che lavora per codificare frammenti di sistemi legislativi tramite la programmazione logica, senza però modellare le molte eccezioni che la legislazione pone in essere;
- "Marcell" (https://marcell-project.eu/), che utilizza AI per migliorare il multilinguismo nei documenti giuridici, ma che poi non include la semantica dei documenti giuridici.

Ognuno di questi progetti è però isolato, in quanto prosegue in una direzione che non mira ad una ricerca comune o ad una struttura giuridica che sia robusta e dinamica come ci si aspetterebbe.
Degno di nota è inoltre il progetto ERC "CompuLaw" (https://site.unibo.it/compulaw/en/project) , concentrato sulla modellazione logico-simbolica e non-simbolica della giurisprudenza con tecniche di l'intelligenza artificiale. Il progetto pone particolare attenzione alla valutazione critica.
Un altro progetto promettente è l'ERC "HyperModeLex" (https://cordis.europa.eu/project/id/101055185) che ha il compito di investigare l'introduzione ed integrazione e definire nel framework regolatorio per l'infosfera ibrida tra normative dirette all'uomo e normative dirette ad entità computazionali, nel rispetto dei processi legislativi e dei principi di teoria del diritto.

2.2 - Tecnologie, strumenti, standard, obiettivi

È obiettivo dello studio "Legal Drafting in the Era of Artificial Intelligence and Digitisation" quello di un framework legislativo supportato dai principi di filosofia del diritto, informatica giuridica, informatica (AI) e linguistica computazionale (NLP) che permetta a rendere legale, autentico e autoritativo il diritto computabile dalla macchina prodotto direttamente dalle istituzioni.
L'approccio utilizzato per raggiungere questi scopi è un framework tecnologico ibrido per l'intelligenza artificiale. Questo approccio è stato scelto a fronte dei maggiori problemi riscontrati con le attuali applicazioni AI non simboliche (ML/DL):

- Un funzionamento privo di logica o semantica che nel contesto giuridico sono essenziali;
- Mancanza di informazioni relative alla struttura del documento;
- Scarsa capacità di interpretazione del testo;
- Evidenti mancanze nella gestione delle citazioni giuridiche, commettendo spesso errori per via del funzionamento di alcuni algoritmi che non hanno modo di valutare correttamente i testi citati;

- Lacuna di parametri temporali quali la validità legale, l'efficacia e l'applicabilità delle norme;
- Mancata integrazione delle annotazioni web di relazioni all'interno delle frasi.

Queste sono le principali ragioni per cui l'architettura realizzata include applicazioni AI simboliche e non simboliche, usando lo standard XML per i documenti giuridici Akoma Ntoso (Akoma Ntoso Version 1.0. Part 1: XML Vocabulary) appartenente alla famiglia degli standard OASIS LegalXML che fonda un corpo annotato comune per applicativi AI robusti.

Akoma Ntoso implementa il principio di "self-containment", ciò vuol dire che tutti i meta dati e la conoscenza sono rappresentate nella stessa struttura logica XML. Questo principio permette ad applicazioni AI di ottenere tutte le informazioni richieste per dedurre nuove informazioni o riuscire a fornire spiegazioni in merito ai metadati giuridici (e.g., validità di un articolo).

L'utilizzo, infine, dei principi "human-in-the loop", "human-on-the-loop", e "human-in-command" sono in risposta alla necessità del dominio legale di prevenire bias, applicare la teoria dell'interpretazione, ed integrare un modello machine-readable insieme al ragionamento giuridico fornito dall'esperto di dominio.

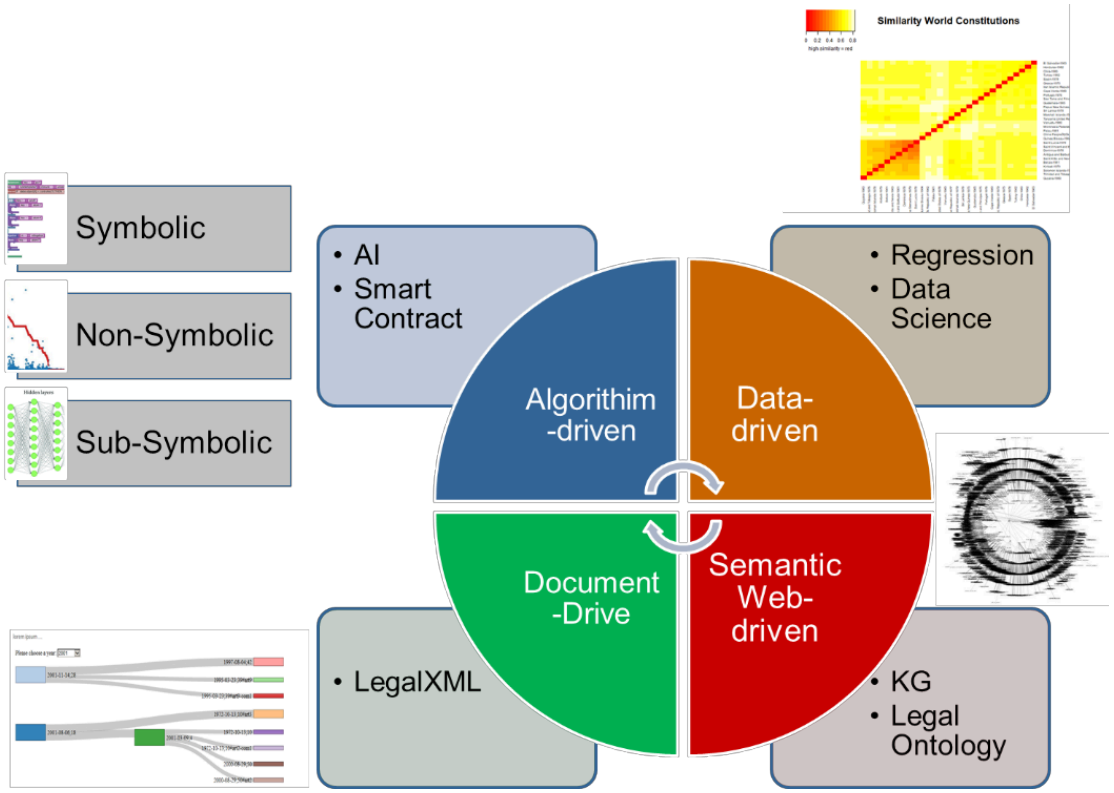

Figura 1 - L'intelligenza artificiale e l'ambito giuridico [PV22] (Palmirani 2023).

2.3 - Il processo legislativo europeo

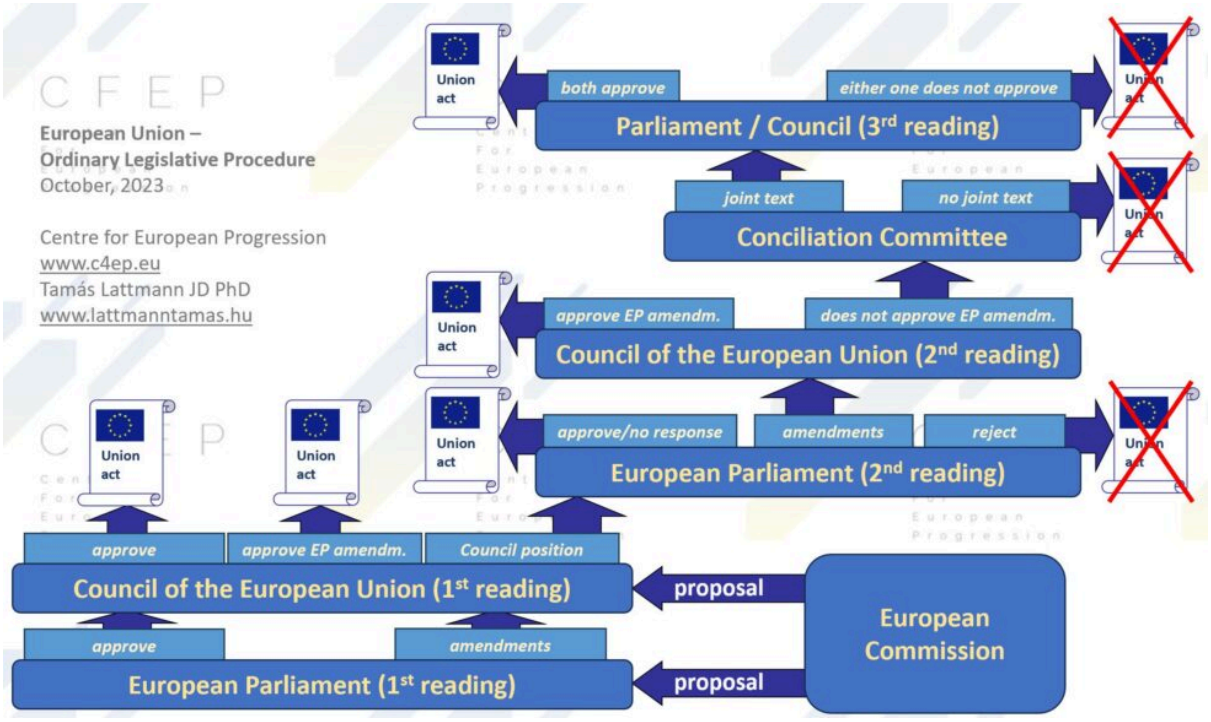

Figura 2 - Step del processo legislativo europeo[3]

Comprendere il modo in cui le istituzioni europee scrivono le leggi è fondamentale per esplorare le problematiche che ne conseguono ed analizzare possibili soluzioni. Innanzitutto, il processo legislativo cerca di trovare un delicato equilibrio tra la sovranità degli stati membri e gli interessi "comuni" dell'intera comunità degli stati membri, dando il giusto peso ai rappresentanti dell'elettorato europeo. Questo significa che le tre istituzioni dell'Unione Europea, ovvero:

1. La Commissione Europea, che rappresenta gli interessi dell'Unione nel complesso ed è responsabile per la creazione di iniziative per le leggi UE;
2. Il Parlamento europeo, che rappresenta i cittadini UE, di fatto vengono eletti proprio da questi ultimi;
3. Il Concilio dell'Unione Europea a livello ministeriale, che rappresenta i governi dei singoli Stati membri.

Il ruolo di queste istituzioni ha subito sostanziali cambiamenti nel corso degli anni, ed il processo propositivo legislativo attuale parte dalla Commissione Europea, che ha il diritto esclusivo di creare proposte per gli organi legislativi. Anche se "l'idea" proviene da

[3] The EU legislative procedure – trying to strike a delicate balance between sovereignty of member states, interests of the whole community, while giving weight to the representatives of the European electorate

uno stato membro, da un membro del Parlamento europeo o da chiunque altro, inclusi i cittadini dell'UE, l'iniziativa ha sempre origine dalla Commissione. L'articolo 225 del Trattato sul funzionamento dell'UE regola il modo in cui il Parlamento europeo può "avviare iniziative" con la Commissione Europea, che non ha alcun obbligo in tal senso. Infine, il Trattato di Lisbona ha introdotto la cosiddetta "Iniziativa dei Cittadini Europei" come un modo per convogliare le iniziative del pubblico (Art. 11 Par. 4 del Trattato sull'Unione Europea), ma, nuovamente, ciò non impone alcun obbligo sulla Commissione.

2.4 - I Principali beneficiari dello studio

Alla luce del processo legislativo europeo, un problema che emerge durante le prime fasi è la scrittura della normativa detta attività di legal drafting.
In particolare, è rilevante sapere come i principali protagonisti della stesura dei documenti siano funzionari della commissione, ovvero professionisti di specifici settori che svolgono una varietà di compiti, tra cui la preparazione di documenti, la gestione delle sessioni interne, l'assistenza ai politici della Commissione e la supervisione delle procedure legislative.
Dati i loro campi di specializzazione variegati, la loro preparazione e conoscenza delle legislazioni è ottima. Tuttavia, durante la scrittura è possibile aspettarsi errori accidentali riguardo la consistenza della forma e dei contenuti con altri documenti scritti in contesti diversi e da altre persone, nonché incongruenze o riferimenti errati ad altri documenti.
Questo studio si concentra nel fornire strumenti di supporto capaci di migliorare il processo di scrittura legislativa per queste figure professionali soprattutto nella stesura dei riferimenti normativi.

Capitolo 3 - NormRefAI assistant

3.1 - Il progetto ed il NormRefAI assistant

Nello sviluppo del progetto sono stati individuati [PV22] diversi possibili componenti per il supporto alle attività di redazione giuridica (LDTS), tra cui:

1. Preambolo - auto completamento del testo della parte giustificativa della proposta legislativa in base al corpus legislativo vigente;
2. Suggeritore di riferimenti normativi- suggeritore di riferimenti normativi dato un iniziale nucleo di dati
3. Sintassi - correzioni a livello sintattico del testo per esempio suggerendo i verbi corretti (e.g. “must” “should”);
4. Redazione corretta - ragionamento giuridico ossia se i riferimenti sono coerenti nel contesto giuridico in cui sono collocati;
5. Controllo delle definizioni - correttezza delle definizioni giuridiche in accordo con l’intero sistema legislativo e il contesto storico;
6. Uso aggiornato dei termini collegati a una policy - uso corretto dei termini basato sulla legislazione vigente e alla policy che si intende perseguire (e.g., climate change).

Il mio contributo si sviluppa nella seconda funzionalità descritta in lista, un suggeritore di riferimenti giuridici mediante strumenti di NLP e funzionalità AI di similarità tematica dei documenti.

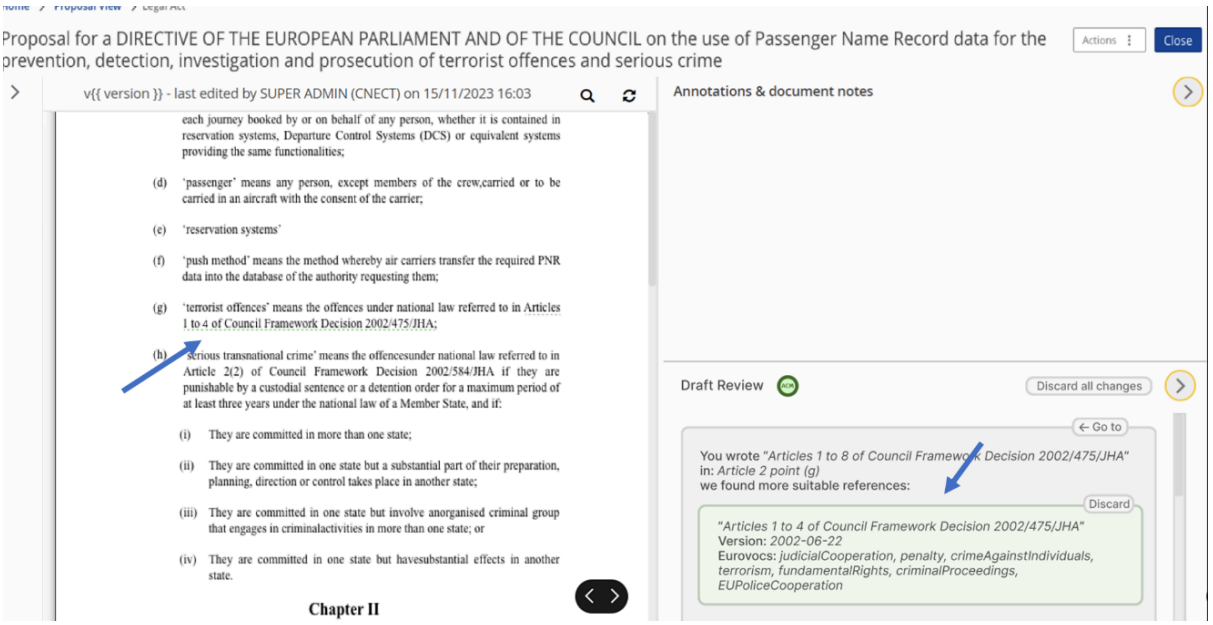

Figura 3 - Mock up dell'assistente di scrittura LEOS del front-end

Il componente è in grado di riconoscere i riferimenti completi ed incompleti presenti in un testo, calcolare il grado di pertinenza tra il documento che si sta scrivendo e il documento a cui si sta facendo riferimento tramite il vocabolario tematico di classificazione semantica EuroVoc[4]. Il NormRefAI assistant permette di completare tale riferimento se necessario e restituire un elemento formalizzato mediante la naming convention di Akoma Ntoso[5] che rappresenta il riferimento con tutti i metadati necessari all'inserimento dello stesso nel documento, insieme ad informazioni utili all'utente finale per la navigazione futura nel Web.

Un esempio pratico di utilizzo segue:

Input:

- Riferimento: "regulation (EU) 406"
- EuroVoc: "communityCertification", "domesticMarket", "freeMoevementOfGoods"

Output:

- Score: 91.72%
- Espressione FRBR: "/akn/eu/act/regulation/2010-04-26/406-2010/eng@/!main"
- Suggerimento: "<ref eId="ref_1" href="/akn/eu/act/regulation/2010-04-26/406-2010/"> regulation 406/2010</ref>"

[4] EuroVoc - EUR-Lex
[5] Akoma Ntoso Naming Convention Version 1.0

- EuroVoc: "communityCertification", "domesticMarket", "freeMovementOfGoods", "hydrogen", "motorFuel", "motorVehicle", "pollutionControl", "technicalStandard"
- Data validità: "2010-06-07"
- Messaggio: "suggestion"

3.2 - Premesse tecnologiche

Alcuni importanti elementi tecnologici sono la base del componente NormRefAI assistant:

1. FormeX (https://op.europa.eu/en/web/eu-vocabularies/formex);
2. Akoma Ntoso (https://docs.oasis-open.org/legaldocml/akn-core/v1.0/os/part1-vocabulary/akn-core-v1.0-os-part1-vocabulary.html);
3. EuroVoc (http://publications.europa.eu/resource/dataset/eurovoc);
4. LEOS (https://joinup.ec.europa.eu/collection/justice-law-and-security/solution/leos-open-source-software-editing-legislation).

Formex

FormeX è lo standard XML adottato sin dal 1998 per rappresentare i documenti dell'European Publication Office per la gestione della Gazzetta Europea EUR-LEX[6]. Il nuovo schema v4 è stato emanato nel 2004 e i documenti sono scaricabili mediante CELLAR[7] (https://op.europa.eu/en/web/cellar).
Nel nostro progetto i documenti in inglese sono stati forniti dall'European Publication Office in Formex v4 dal 2010 al 2020.

Akoma Ntoso - LegalDocML

Akoma Ntoso è uno standard internazionale approvato da OASIS nel 2018 dal Technical Committee LegalDocML [PV18]. AKN è un vocabolario XML per l'annotazione di documenti giuridici, emanato nella sua prima versione grazie alle Nazioni Unite nel 2008 mediante il progetto "*Africa i-Parliament Action Plan*" (https://publicadministration.desa.un.org/projects/raf08x01-africa-i-parliaments-action-plan).

[6] EUR-Lex
[7] Data reuse - EUR-Lex

Le istituzioni europee hanno recepito Akoma Ntoso fra gli standard da utilizzare ufficialmente nella gestione documentale giuridica per favorire l'interoperabilità fra i diversi organismi coinvolti nell'informazione normativa e giuridica in generale.
Hanno creato una personalizzazione di Akoma Ntoso chiamata AKN4EU (https://op.europa.eu/en/web/eu-vocabularies/akn4eu).
Nel nostro progetto un altro gruppo ha convertito i documenti Formex in AKN dotati di classificazione EuroVoc.

EuroVoc

EuroVoc è il thesaurus multilingue e multidisciplinare dell'UE. Comprende parole chiave, organizzate in 21 settori e 127 sottosettori, utilizzate per descrivere il contenuto dei documenti in EUR-Lex. Eurovoc comprende 26 lingue, quelle ufficiali delle istituzioni europee.
Nel nostro progetto tutti i documenti sono dotati di classificazione tematica EuroVoc.

LEOS

L'editor LEOS è un web editor specializzato per la creazione e la modifica di documenti della Commissione europea che produce documenti in formato AKN4EU. Offre una vasta gamma di funzionalità che lo rendono adatto a utenti che hanno come obiettivo il legal drafting[8]. LEOS è stato creato da un gruppo informatico interno dalla DG Informatics ed è un progetto portato avanti in diverse direzioni, inclusa la personalizzazione per gli stati membri dell'Unione Europea.

3.3 - Utilizzi in LEOS

Il fine ultimo dell'implementazione di questo componente NormRefAI assistant risiede nell'assistenza di un particolare utente finale, identificato come un funzionario della Commissione europea, ovvero un professionista del settore che, sebbene non operi direttamente come legislatore trarrà sicuramente vantaggio dall'utilizzo del componente in questione. Questi professionisti, pur essendo adeguatamente preparati nel loro campo di competenza, potrebbero aver bisogno di assistenza mediante strumenti informatici durante la redazione dei testi legislativi che risultano complessi (e.g., allegati, trattati, accordi) e che vedono il passaggio in molti uffici secondo un workflow articolato.

[8] LEOS - Open Source software for editing legislation | Joinup

Pertanto, uno strumento che confermi, segnali o integri i riferimenti normativi fornirà loro un valido supporto nella redazione dei documenti, migliorando significativamente la loro esperienza utente e garantendo una maggiore precisione e affidabilità nel processo decisionale e nella produzione dei testi legislativi.
Per questo motivo il componente è integrato all'interno di LEOS.
Il componente NormRefAI assistant è inoltre altamente accessibile grazie alla possibilità di essere contattato tramite un API RESTful, elemento approfondito nel paragrafo successivo.
Dunque, il componente per riferimenti normativi mira a fornire un livello di supporto provvedendo:

- Una consistenza nella forma della stesura dei riferimenti anche rispetto ai diversi periodi storici;
- Un'analisi della correttezza di riferimenti completi;
- Suggerimenti su riferimenti incompleti o poco coerenti;
- Informazioni utili per verificare l'eventuale pertinenza tematica del riferimento rispetto al documento che si sta scrivendo mediante le keyword di EuroVoc.

3.4 - Casi d'uso

I casi d'uso possono essere molteplici ma ne abbiamo identificati quattro dove il componente può essere integrato:

1. in LEOS a supporto della scrittura delle proposte legislative della Commissione Europea;
2. nei sistemi informativi degli stati membri quando questi devono integrare nella loro legislazione domestica i riferimenti normativi della legislazione europea,
3. nella redazione delle sentenze per aiutare i giudici a citare correttamente gli atti legislativi europei;
4. integrato nei motori di ricerca per poter raffinare le query.

In questo elaborato ci si concentrerà sul primo caso d'uso.

Capitolo 4 - Architettura del normRefAI assistant

4.1 - Architettura ad alto livello

L'architettura del componente per riferimenti si divide in 3 sotto componenti:

- Un database MySQL;
- Un'istanza Server NodeJS;
- Un applicativo scritto in Python.

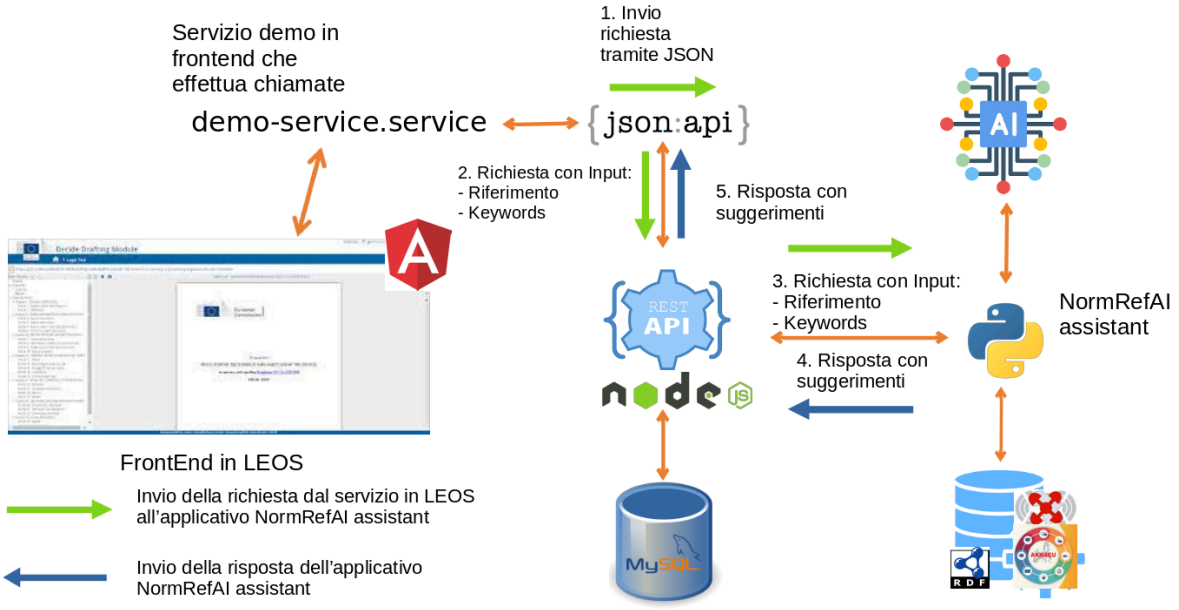

Figura 4 - Schema di richiesta e risposta da frontend LEOS

Mentre l'applicativo Python rappresenta il cuore del componente, i restanti due elementi forniscono metodi di comunicazione interni necessari all'esecuzione di tutte le funzionalità.
Vediamoli in particolare.

1. Il database

Questo sotto componente è utilizzato internamente dall'applicativo Python, comunicando tramite NodeJS per accedere ad informazioni sui documenti, in particolare

su un campo necessario a elaborare la coerenza tra le keyword ed il documento. Lo scopo del database è fornire al servizio uno strumento affidabile per verificare con sicurezza che i suggerimenti in risposta corrispondano a documenti realmente esistenti e soprattutto validi al tempo di scrittura.
Molte delle informazioni del documento ritornate dal componente in assistenza dell'utente finale provengono dallo schema di questo database.

2. Il server

Il sotto componente scritto in NodeJS ha una triplice funzionalità:

- Comunica con il database MySQL e restituisce all'applicativo Python le informazioni appositamente richieste gestendo internamente le query;
- Fornisce un API RESTful nella quale è possibile contattare il database ed il componente, espandibile per contattare altri futuri componenti in sviluppo;
- Provvede all'utente finale una documentazione per istruire sul funzionamento delle varie chiamate disponibili nel servizio fornito tramite un'interfaccia swagger.

Alla luce di queste utilità, il server NodeJS assume importanza soprattutto all'interno dello studio portato avanti dall'università di Bologna, assumendo il ruolo di un punto centrale per interfacciarsi con diversi schemi, database, componenti parte del progetto che necessitano moderazione nel loro accesso.

3. l'applicazione

Punto chiave del componente e cuore della tesi, i file Python che compongono il terzo ed ultimo sotto componente, sono i principali responsabili di fornire all'utente finale una risposta significativa e coerente sfruttando la comunicazione con il server NodeJS e le informazioni presenti nel Database MySQL ed eXist.
Esso comprende anche dei file responsabili per la creazione e gestione del database MySQL, dove vengono pre calcolati gli indici di ogni documento presenti nel database eXist di riferimento, e comprendono l'opzione di creare e tenere aggiornati gli schemi presenti.

4.2 - Il database

Punto di appoggio per il sotto componente Python per la raccolta di informazioni su documenti. Presenta due schemi al suo interno, convenzionalmente chiamati "Keywords" e "Documents". La loro struttura si presenta come segue:

4.2.1 “Documents” schema

Nome colonna	Tipo
FRBREXPRESSIONTHIS	varchar(100) NOT NULL
FRBRWORKTHIS	varchar(100) NOT NULL
ValidityDate	date NOT NULL
DocDate	date NOT NULL
DocYear	int NOT NULL
DocNumber	int NOT NULL
DocType	varchar(15) NOT NULL
AliasAKNShort	varchar(100) NOT NULL
AliasCELEX	varchar(20) NOT NULL
DocIndex	blob NOT NULL

Questo schema contiene la maggior parte delle informazioni necessarie per il componente ad effettuare una ricerca sui possibili documenti da suggerire, elenco:

- “FRBREXPRESSIONTHIS” corrisponde alla chiave delle voci dello schema, rappresenta il valore del campo “FRBRExpression” del documento, usato in AKOMA NTOSO come identificativo;
- “FRBRWORKTHIS”, rappresenta il valore del campo “FRBRwork” del documento, utile all’identificazione tramite apposito URI a cui corrisponde;
- “ValidityDate”, contiene l’ultima data di validità del documento trovata;
- “DocDate”, è la data di pubblicazione riportata nel documento e corrispondente data contenuta nel campo “FRBREXPRESSIONTHIS”;
- “DocYear”, corrisponde all’anno di pubblicazione;
- “DocNumber”, rappresenta il numero seriale del documento, anch’esso contenuto nel campo “FRBREXPRESSIONTHIS”;
- “DocType”, può assumere il valore di “regulation”, “decision” oppure “directive”, rappresenta il tipo di documento;

- “AliasAKNShort”, contiene un URI alternativo ad “FRBREXPRESSIONTHIS”, semplificato per essere identificato dai collegamenti presenti nei riferimenti;
- “AliasCELEX”, rappresenta il codice univoco CELEX del documento;
- “DocIndex”, corrisponde ad un vettore avente 300 dimensioni, utilizzato come metrica per la vicinanza tra le keyword del documento in scrittura e le keyword del documento rappresentato da questo elemento del database.

4.2.2 “Keywords” schema

Nome colonna	Tipo
Keyword	varchar(100) NOT NULL
Vector	blob NOT NULL

Questo schema è utilizzato a runtime dal componente python per calcolare la media di tutti i vettori corrispondenti alle keyword in ingresso nel componente, che verrà successivamente utilizzato nell’assegnare uno score ad ogni suggerimento.
I campi di questo schema corrispondono come segue:

- “Keyword”, chiave delle voci dello schema, corrisponde ad una keyword manipolata in modo che rispetti il camel case e che non presenti caratteri speciali all’infuori di quelli alfanumerici;
- “Vector”, corrisponde ad un vettore avente 300 dimensioni che a sua volta corrisponde ad una media di tutte le singole parole che compongono la keyword a cui appartiene, pre-calcolato da un modello AI Word2Vec noto.

4.2.3 Popolazione del database e calcolo dei vettori

La popolazione del database avviene tramite due file Python. Uno di questi si occupa di creare un file di testo contenente tutti i vettori e le keyword formalizzate per rispettare il formato in cui sono inseriti nel database.
Questo file Python, denominato “create_eurovoc_txt.py” ha bisogno di un file .vec, output di un modello AI Word2Vec, per ricavare i vettori che poi andranno inseriti nel file di testo.

Attualmente il modello scelto è “crawl-300d-2M-subword” (https://fasttext.cc/docs/en/english-vectors.html), creato sulla base di “due milioni di vettori di parole addestrati con informazioni sulle sotto parole su Common Crawl (token da 600 miliardi)” [TMG+17].
L’applicativo Python ha due opzioni principali di calcolo del vettore associato alla singola keyword:

- Se si tratta di una keyword composta da una parola, esso inserisce il vettore corrispondente nel file di testo insieme alla corrispettiva parola;
- Se la keyword è composta da più di una parola, vengono trovati tutti i vettori delle parole che lo compongono, e viene effettuata una media tra tutti i vettori trovati. Il risultato viene infine aggiunto al file di testo con la corrispettiva keyword nel formato richiesto dal database.

Il secondo file Python, denominato “create.py”, si occupa di connettersi con il database MySQL, crearlo insieme agli schemi se necessario, e popolarli. Per primo viene creato lo schema “Keywords” e popolato tramite il file txt generato precedentemente, dove non è richiesto nessun calcolo aggiuntivo se non quello di trasformare il vettore in un bytes-object. Successivamente viene creato lo schema “Documents” e popolato a sua volta con tutte le informazioni relative ai documenti, ottenute tramite una query ad un’istanza eXistDB, contenente i documenti con le informazioni necessarie per riempire i campi dello schema e calcolare il vettore corrispondente ad ogni documento. I documenti presenti nell’istanza eXistDB sono ricavati a partire da documenti presenti nel database CELLAR fornito dall’Unione Europea per consulto ufficiale. Nello schema dei documenti, il calcolo del vettore avviene con l’analoga operazione di media tra vettori, prendendo come operandi i vettori di ogni keyword a cui il documento fa riferimento dal primo schema. Il risultato dell’operazione è un vettore che rappresenta in modo significativo, la posizione nello spazio vettoriale del documento in base alle keyword a cui appartiene. Ciò ci permette di poter riconoscere, tramite questo vettore, la distanza tematica tra un documento ed un qualsiasi insieme di EuroVoc rilevanti.

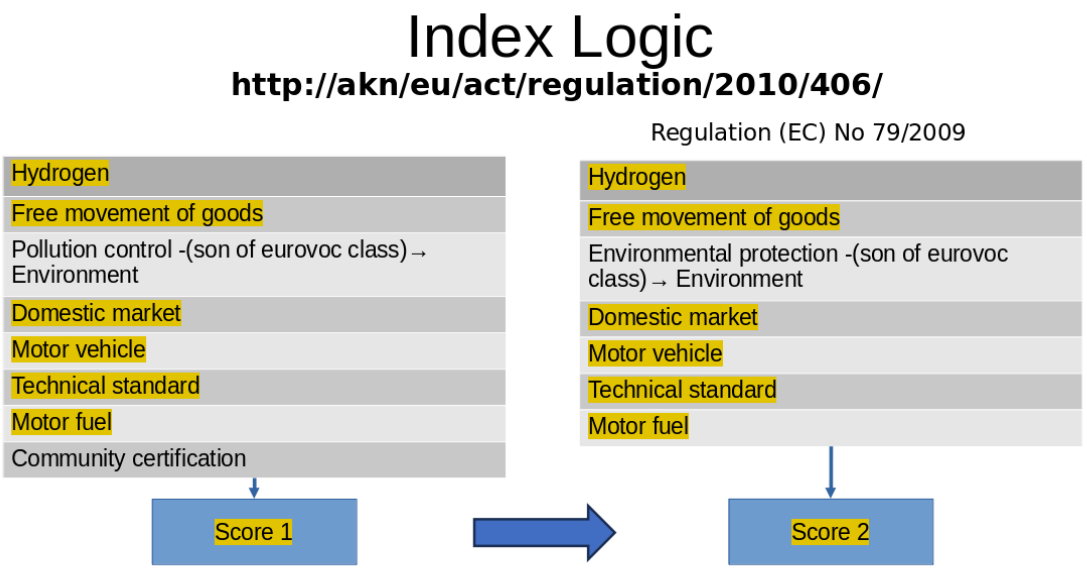

Figura 5 - Logica di indicizzazione dei documenti

Nella figura 5 possiamo osservare un esempio grafico human-readable del significato dello "score" attribuito ad ogni documento. Esso corrisponde all'insieme delle keyword presenti in ogni documento, che viene poi usato nel confronto per ogni caso. Nella figura è presente un esempio tra le keywords del documento "regulation 406/2010" e "regulation (EC) No 79/2009".

4.3 - Il server

La necessità di creare un server NodeJS nasce primariamente dal bisogno di avere disponibile un'interfaccia che tutti possano usare ed espandere su una base solida, dinamica e ben documentata.
A questo proposito, l'idea del servizio NodeJS è fornire dal lato utente un accesso tramite API RESTful ai componenti in sviluppo, così che possano essere usati in modo libero da più fonti dato il diffusissimo protocollo di comunicazione. Il server risponde anche alla necessità di standardizzare gli accessi al database MySQL, ponendo controlli di sicurezza necessari per l'integrità di quest'ultimo.

4.3.1 - Configurazione

È possibile configurare alcune impostazioni del server tramite un file JSON denominato "config.json". In particolare, si puo' definire se il server deve offrire una connessione con

protocollo HTTP oppure HTTPS, le impostazioni di connessione al server MySQL ed il percorso dei componenti da eseguire.
Il server gira su cluster, generando un processo figlio all'avvio ed un nuovo processo figlio ogni volta che quello precedente non è più in esecuzione, creando così un servizio che anche in caso di un errore non gestito possa continuare ad essere disponibile.

4.3.2 - Documentazione API

Attualmente, tramite il percorso "/api-docs" è possibile accedere alla documentazione sulle possibili chiamate gestite dal server.

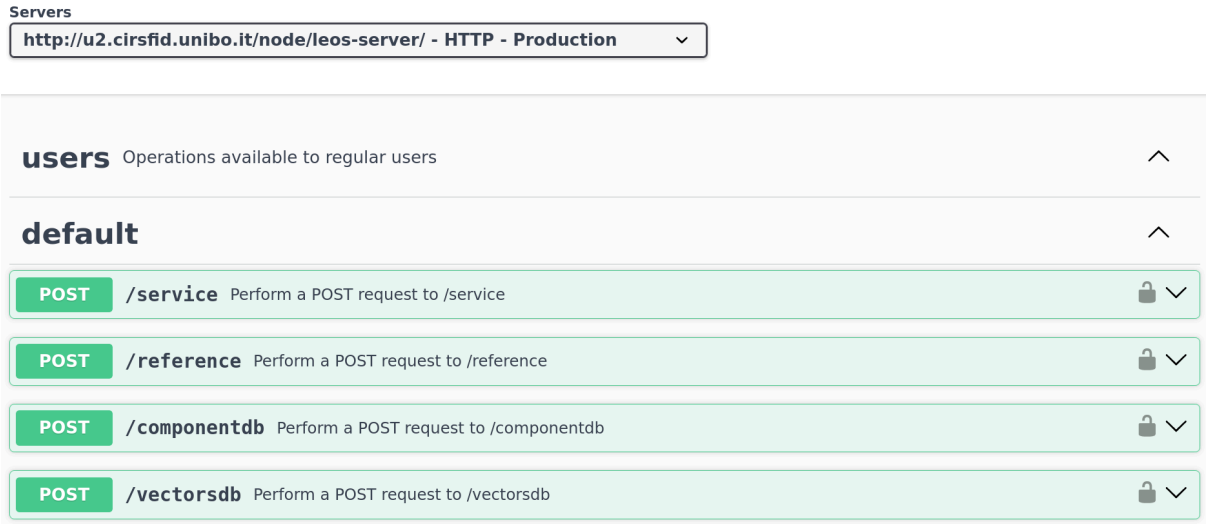

Figura 6 - Pagina api-docs del server NodeJS

Questa pagina è stata creata tramite l'utilizzo di Swagger UI (https://swagger.io/tools/swagger-ui/), un tool per visualizzare ed interagire con le chiamate disponibili che permette a sviluppatori back end di tenere facilmente aggiornate tutte le informazioni necessarie per sviluppatori front end oppure consumatori.

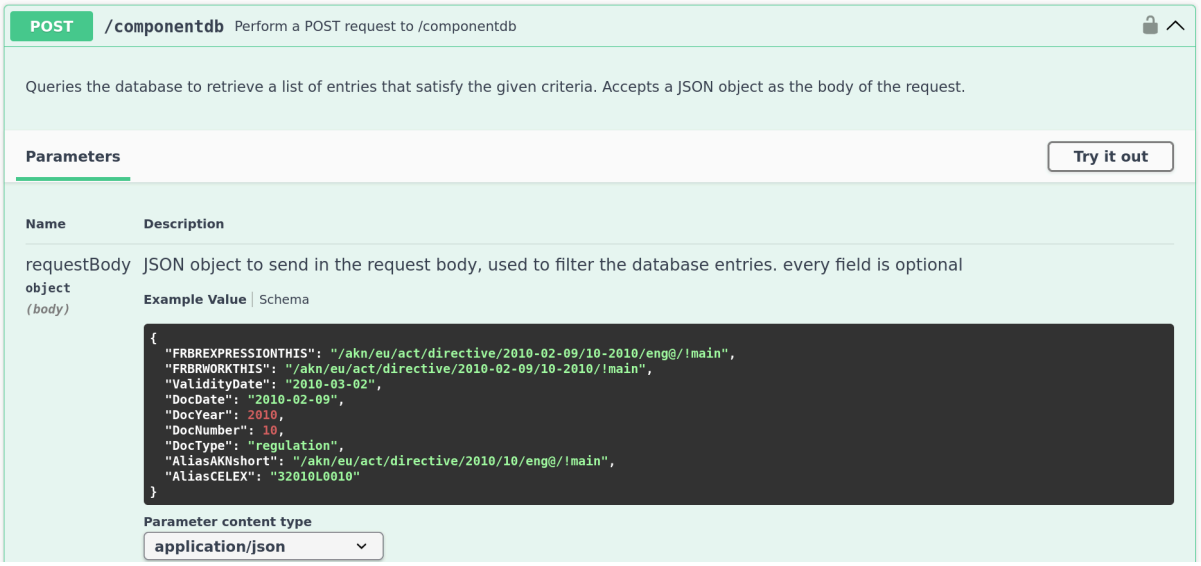

Figura 7 - Chiamata POST al database MySQL attraverso il server

La documentazione contiene esempi di comunicazione, schemi dei dati in risposta, e l'opzione di testare le funzioni tramite curl direttamente all'interno della pagina.

4.3.3 - Struttura concettuale

All'interno del progetto, l'architettura complessiva del servizio è stata definita come in figura:

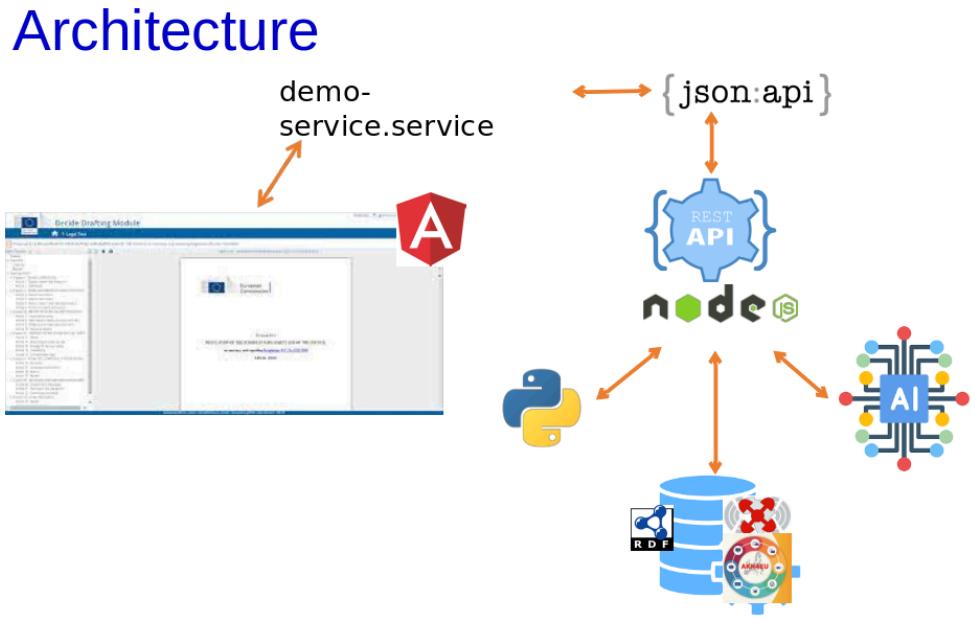

Figura 8 - Architettura del servizio in sviluppo

L'architettura scelta rende il server NodeJS e l'API il punto centrale della comunicazione tra i vari elementi necessari al funzionamento del servizio.

4.3.4 - Struttura concreta

I file sorgente del server comprendono:

- "app.js", il principale file dove viene inizializzato il server e che si occupa di generare processi figli quando richiesto, oltre che definire la pagina di documentazione;
- "server.js", una raccolta delle dipendenze del server e setup di alcune variabili utilizzate globalmente dall'applicazione;
- "config.json", file di configurazione descritto in precedenza;
- "JSON", cartella contenente i documenti in formato json necessari ad alcune chiamate per fornire il loro servizio;
- "src", cartella contenente le varie routes del server, organizzati in maniera modulare, insieme alla documentazione usata per la pagina "api-docs". all'interno della cartella abbiamo:
    - "mysql_routes.js", contenente le chiamate che gestiscono i servizi "componentdb" e "vectorsdb". Come il nome implica, si tratta di chiamate che contattano il database MySQL;
    - "reference_routes.js", al momento contenente la chiamata "reference", in futuro conterrà le chiamate che gestiscono i vari componenti al momento non disponibili, insieme all'attuale componente per riferimenti giuridici.
    - "service.js", un servizio mock-up contenente la chiamata "service", è stata necessaria per effettuare test di connessione tra il server ed eventuali componenti esterni;
    - "swagger.yaml", contenente la documentazione utilizzata nella pagina "api-docs" in formato YAML.

4.4 - L'applicazione

4.4.1 - Struttura concettuale

L'infrastruttura del programma segue un complesso processo che coinvolge più volte gli altri due sotto componenti appena descritti. La struttura concettuale può essere sintetizzata con la seguente figura.

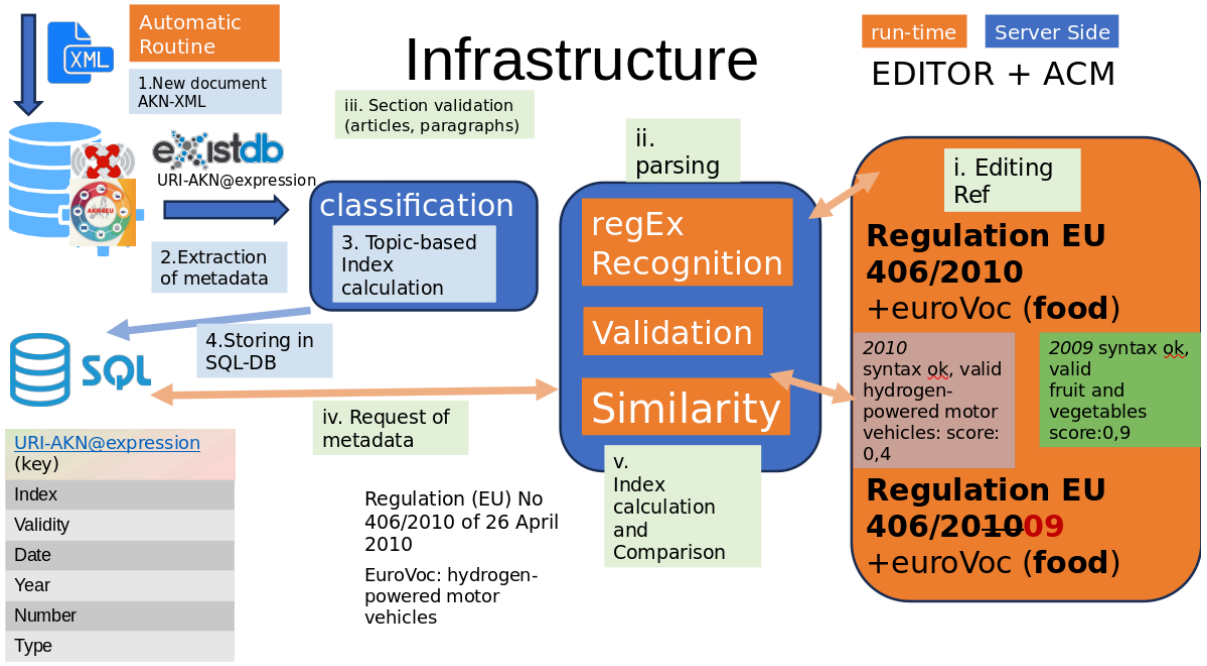

Figura 9 - Infrastruttura dell'applicativo Python

L'applicativo a livello concettuale segue due processi distinti: la creazione ed aggiornamento del database MySQL, e la creazione dei suggerimenti da restituire in output.
Seguendo il processo riguardante l'aggiornamento del database, esso corrisponde ad una routine automatica che si avvia nel momento in cui nel database eXist viene creato o aggiunto un nuovo documento. Il programma effettua una query ad eXistDB per ricavare i metadati necessari al calcolo dell'indice del documento ed i dati richiesti di ogni campo interno del database MySQL.
Successivamente alla risposta del database eXist, viene calcolato l'indice e inserito insieme al resto dei dati rilevanti all'interno del database MySQL.
Spostandosi invece al processo principale, come prima cosa L'applicativo effettua un parsing del riferimento ricevuto tramite espressioni regolari, procedendo ad una tokenizzazione degli elementi che serviranno nella richiesta verso il database MySQL.
In seguito alla risposta, L'applicativo valuta lo score di similarità con ogni potenziale documento da suggerire, ne genera il suggerimento in formato AKN-friendly, e lo aggiunge ai risultati che verranno restituiti in output.

4.4.2 - Struttura concreta

L'applicativo è composto dai file seguenti:

- “config.py”: contiene molte variabili utilizzate globalmente da altri file, permettono di modificare diverse dipendenze dell’applicativo tra cui:
    - Variabili Flask, ovvero:
        - “FLASK_HOST”: l’host da cui poter contattare il componente nel caso si decida di usare la modalità server interna al programma;
        - “FLASK_PORT”: la porta in ascolto;
        - “FLASK_DEBUG”: la modalità di debug del server una volta attivato.
    - Variabili eXist-db, ovvero:
        - “USE_EXIST”: indica se utilizzare l'istanza di eXist-db come metodo principale di interrogazione. Se impostato su False, verrà utilizzato il server MySQL e eXist-db verrà utilizzato solo per il recupero degli Eurovoc appartenenti ad un possibile documento suggerito;
        - “EXISTDB_SERVER_URL”: specifica l'URL dell'istanza di eXist-db;
        - “EXISTDB_ROOT_COLLECTION”: specifica la collezione dell'istanza di eXist-db da interrogare;
        - “EXISTDB_SERVER_USER”: specifica il nome utente dell'istanza di eXist-db;
        - “EXISTDB_SERVER_PASSWORD”: specifica la password dell'istanza di eXist-db;
        - “AKN_NAMESPACE”: specifica il namespace di Akoma Ntoso.
        - “EXISTDB_TIMEOUT”: specifica il timeout per la connessione a eXist-db;
        - “EXISTDB_POPULATE_LIMIT”: specifica il numero massimo di documenti da interrogare contemporaneamente;
        - “EXISTDB_POPULATE_SLEEP”: specifica il tempo di attesa tra ogni interrogazione in secondi.
    - Variabili server MySQL, ovvero:
        - “NODE_HOST”: specifica l'URL del nodo MySQL Server;
        - “SQL_POST_REQUEST”: specifica l'URL per la richiesta POST al database del componente MySQL Server;
        - “SQL_VEC_POST_REQUEST”: specifica l'URL per la richiesta POST al database dei vettori del componente MySQL Server;

        - “SQL_SERVER”: specifica il nome del server MySQL;
        - “SQL_SERVER_PORT”: specifica la porta del server MySQL;
        - “SQL_DATABASE”: specifica il nome del database MySQL;
        - “SQL_TABLE”: specifica il nome della tabella dei documenti;
        - “SQL_VEC_TABLE”: specifica il nome della tabella dei vettori;
        - “SQL_USER”: specifica il nome utente per la connessione al server MySQL;
        - “SQL_PASSWORD”: specifica la password per la connessione al server MySQL;
    - Variabili Word2Vec:
        - “VECTORS_PATH”: specifica il percorso del file .vec di fasttext;
        - “TXT_PATH”: specifica il percorso del file di output .txt.
    - Variabili varie del componente:
        - “MINIMUM_SCORE”: specifica la soglia che spinge il componente a cercare documenti diversi dopo la prima ricerca accurata;
        - “MAX_WORKFLOWS”: specifica il numero massimo di flussi di lavoro;
        - “VECTOR_SIZE”: Specifica la dimensione dei vettori nel modello. Questo valore non dovrebbe essere modificato a meno che il modello stesso cambi;
        - “CONDITIONAL_DICT”: Un dizionario di condizioni per la query dinamica di eXist-db. Questo dizionario è utilizzato internamente dal programma e non dovrebbe essere modificato manualmente a meno che non si apportano ulteriori modifiche all’applicativo principale;
        - “SQL_TABLE_STRUCTURE”: Specifica la struttura della tabella SQL utilizzata dall’applicativo. Modificare questa struttura potrebbe compromettere il funzionamento;
        - “SQL_VEC_TABLE_STRUCTURE”: Specifica la struttura della tabella SQL utilizzata per i vettori. Anche questa struttura non dovrebbe essere modificata manualmente per evitare problemi di compatibilità.

- “queries.py”: contiene funzioni che generano query per eXist-db e MySQL tramite parametri appositi, creando un sistema dinamico di creazione delle query in base ai valori richiesti;
- “ref_component.py”: si tratta dell’applicativo principale che genera e classifica i suggerimenti che verranno mostrati all’utente finale;
- “regular_expressions.py”: è una raccolta di espressioni regolari (regular expressions) scritte in parte da Michele Corazza, vengono utilizzate nel processo di tokenizzazione del riferimento;
- “all_modules.txt” & “requirements.txt”: file che raccolgono i moduli python necessari per lo sviluppo ed il deploy rispettivamente;
- “server.py”: permette, attraverso il modulo python Flask, di rendere accessibile il programma attraverso una chiamata POST;
- “vectors.txt”: l’output dell’esecuzione di “create_eurovoc_txt.py”;
- “sql”: una cartella contenente tutti i file relativi alla manutenzione del database MySQL, tra cui:
    - “create.py”: crea il database inserendo gli schemi e aggiungendo i dati rilevanti, si tratta di uno dei file responsabili del pre calcolo dei vettori;
    - “create_eurovoc_txt.py”: crea il file di testo contenente quel che poi saranno i dati presenti nella tabella dei vettori all’interno del database MySQL;
    - “doc_index_text.py”: si tratta di una raccolta di funzioni di test per verificare la correttezza dei vettori calcolati;
    - “update.py”: capace di aggiungere entry in entrambe le tabelle del database. Insieme a “create.py”, è capace di calcolare i vettori nella tabella dei documenti.

4.4.3 - Flusso di esecuzione della popolazione del database MySQL

Il flusso vero e proprio di esecuzione parte dalla generazione del file “vectors.txt”, così da poter spostare il lungo e dispendioso processo di pre calcolo dei vettori al di fuori del processo di popolazione del database MySQL.

Il file “create_eurovoc_txt.py” ricava la classificazione EuroVoc dal database eXist, li trasforma in keyword e ne ricava i vettori, infine facendo una media di tutti i vettori trovati per ricavare il vettore corrispondente ad ogni keyword.

Nel codice, “og_keyword” corrisponde ad una keyword, ad esempio “freedomOfMovement”.

“keyword_list” corrisponderà invece ad un array in questa forma: [“freedom”, “of”, “movement”], e corrisponde alla lista di parole che verranno associate a singoli vettori, per poi diventare una media rappresentante la keyword nella sua interezza.

```
          keyword = re.sub(r"([A-Z])", r" \1", keyword)
          keyword = keyword.lower()

          keyword_list = keyword.split(" ")
          for word in keyword_list:

               if word in keywords_not_found:
                    continue

               vector = load_single_vector(vector_path, word)
               if vector is not None:
                    vectors_loaded =+ 1
                    total_vector += vector
               else :
                    keywords_not_found.add(word)


          # get the average
          total_vector /= vectors_loaded

          vector_exists = not np.allclose(total_vector,
zeros_vector, atol=1e-8) and not
np.any(np.isnan(total_vector))
```

```
          # keyword wasn't found in the vectors
          if not vector_exists:
               print(f"could not find vector for { keyword
}")
```

```
                keywords_not_found.add(keyword)
                total_distinct_keywords_misses += 1
            # keyword exists and it's not already in the txt,
so add it
            else :
                print(f"found { keyword }")
                keywords_to_add[og_keyword] = total_vector
```

Il vettore viene salvato nel dizionario di Eurovoc da aggiungere nel txt con chiave corrispondente alla keyword in Camel case.

Nonostante non ci sia necessariamente bisogno quindi di ottimizzare il codice della generazione dei vettori, delle misure per ridurre il tempo di esecuzione sono state implementate, tra cui:

1. Un set di eurovoc e parole singole non presenti nel file .vec, che previene la ricerca di stringhe nel modello. Sapendo che verranno cercati attraverso due milioni di elementi, se sappiamo che la ricerca è stata effettuata almeno una volta, non c'è bisogno di cercare gli stessi elementi più volte.
2. Un controllo su tutti gli elementi già inseriti nel dizionario che andrà a popolare il file .txt in output, per evitare duplicati e ricerche inutili.

```
            # keyword exists but it's already in the txt, so
skip it
            if og_keyword in keywords_to_add.keys():
                print(f"{ keyword } already in txt")
                continue

            # keyword already searched and not in vectors, so
skip it
            if keyword in keywords_not_found or og_keyword in
keywords_not_found:
                print(f"{ keyword } already searched, not in
vectors, skipping")
                continue
```

Il file “update.py” segue una logica simile per la creazione dei vettori, e segue la medesima routine di popolazione del singolo elemento nello schema richiesto rispetto a “create.py”.

Procedendo verso il processo di creazione del database, lo schema di esecuzione può essere sintetizzato come in figura.

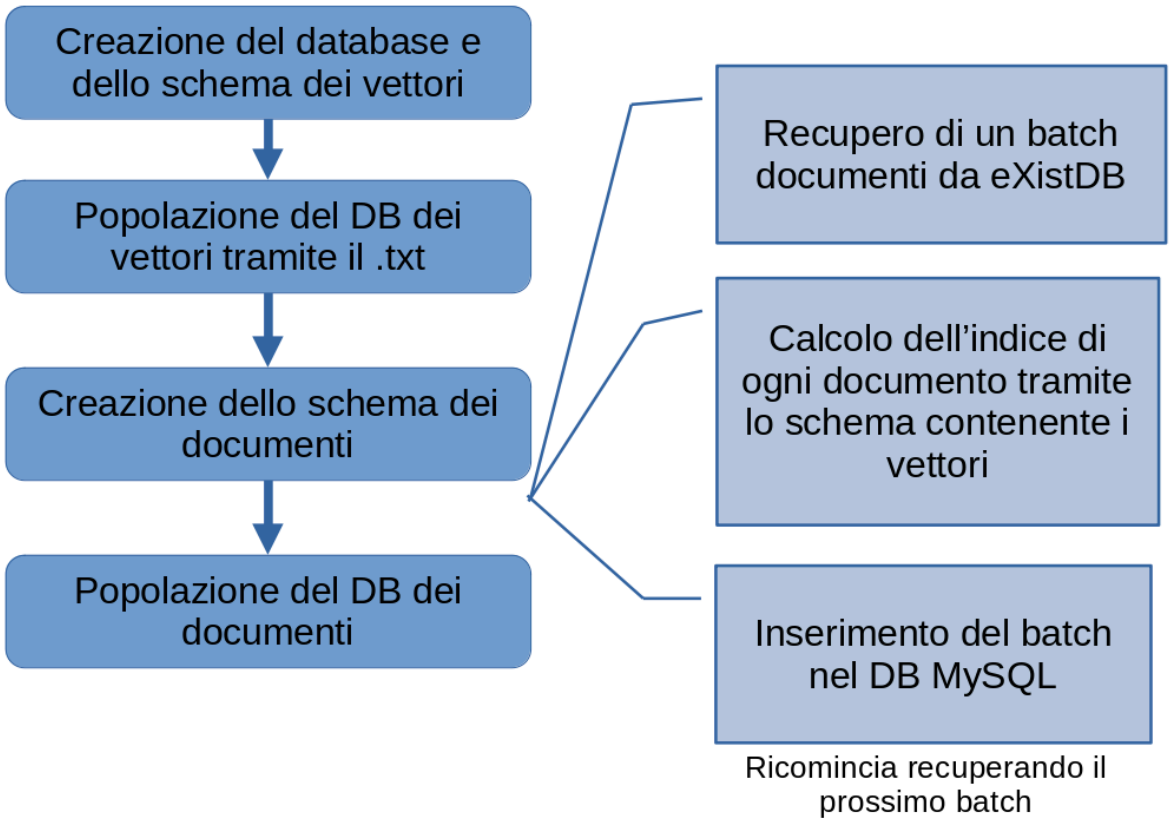

Figura 10 - Infrastruttura dell’applicativo per la creazione del database

La funzione che si occupa del calcolo dell’indice del documento è la seguente:

```
def get_doc_vector(sql_cursor, keywords):
     sql_cursor.execute(queries.sql_get_vectors(keywords))
     rows = sql_cursor.fetchall()

     return np.mean([np.frombuffer(row[0], dtype=np.float64)
for row in rows], axis=0)
```

Essa riceve in input il cursore SQL per richiedere il vettore corrispondente di ogni keyword presente nella lista “keywords”. Una volta prese, restituiamo la media di tutti i vettori, convertiti da oggetti bytes nuovamente in oggetti np.array float64.

4.4.4 - Flusso di esecuzione principale

Il flusso principale di esecuzione include due iterazioni principali, uno interno all’altro.

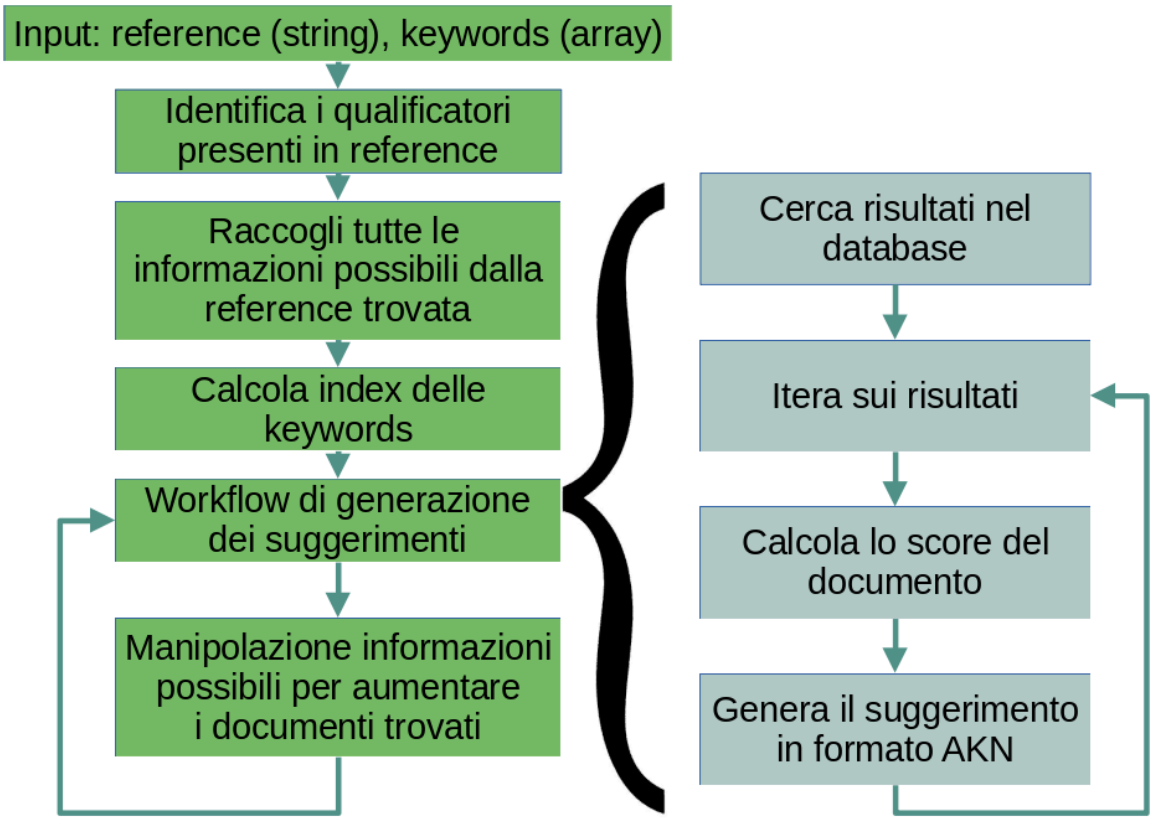

Figura 11 - Infrastruttura dell’applicativo Python principale

Nella prima fase del processo, la stringa viene fatta passare per una funzione chiamata “get_valid_qualifiers”, che si occupa di identificare ogni possibile qualificatore presente nel riferimento (eg. “articolo x, paragrafo y”). Questa classificazione è necessaria per assegnare correttamente l’URI del riferimento includendo qualunque qualificatore citato all’interno del suggerimento. I qualificatori vengono riconosciuti attraverso le espressioni regolari presenti nel file “regular_expressions.py”.
Nella seconda fase prendiamo il cuore del riferimento e ne estrapoliamo le informazioni dal contesto. In particolare, il codice cerca di identificare:

1. Il tipo di documento che stiamo cercando (decision, directive, regulation);

2. L'anno del documento;
3. Il numero sequenziale del documento.

Anche in questa fase l'utilizzo delle espressioni regolari sopra citate è essenziale per catalogare le informazioni trovate: L'applicativo gestisce le informazioni trovate in maniera diversa quando capisce di avere in input un riferimento incompleto.
Nel caso di un riferimento incompleto, il programma tenterà di distinguere l'anno dal numero sequenziale, e nel peggiore dei casi inserirà più ricerche possibili per la query al database. Queste ricerche scambiano i valori ambigui tra anno e numero sequenziale, così da essere sicuri di poter suggerire sempre il documento corretto anche in casi di ambiguità di queste due variabili.

```
   # get the numbers from the reference
     numbers =
regular_expressions.all_numbers_regex.findall(incomplete_refe
rence)

     if len(numbers) > 1:
     # found multiple numbers, take the first two as possible
year and sequential number, the rest is ignored
     gathered_info['year'] = numbers[0]
     gathered_info['sequential_number'] = numbers[1]

     gathered_info_list.append(gathered_info.copy())

     # search for the second case history
     gathered_info['year'] = numbers[1]
     gathered_info['sequential_number'] = numbers[0]

     gathered_info_list.append(gathered_info.copy())
     # this way we eliminate ambiguity by searching both case
studies in the db, as recognizing
     # a year from a sequential number is unreliable
     elif len(numbers) == 1:
     # found only one number, take it as possible year or
sequential number
     gathered_info['year'] = numbers[0]
     gathered_info['sequential_number'] = None
```

```
    gathered_info_list.append(gathered_info.copy())

    # search for the second case history
    gathered_info['year'] = None
    gathered_info['sequential_number'] = numbers[0]

    gathered_info_list.append(gathered_info.copy())

    # no numbers are found, append what we have and return
    if gathered_info_list is None or len(gathered_info_list)
== 0:
    gathered_info_list.append(gathered_info.copy())

    print("gathered_info_list:", gathered_info_list)
    return gathered_info_list
```

La fase successiva riguarda il calcolo dell'indice tramite gli Eurovoc passati in input (chiamati keywords per semplicità all'interno del programma). Questo passaggio è gestito dalla funzione "get_keywords_index":

```
def get_keywords_index(keywords):
    json_keywords = {'keywords': keywords}
    # perform a POST request to the server to get the
keywords index
    response = requests.post(config.SQL_VEC_POST_REQUEST,
json=json_keywords)

    if response.status_code != 200:
    print("Error in the vector POST request")
    return None

    response = response.json()

    all_vectors = []

    for i in range(len(response)):
    # convert the blob object into a numpy array
    bytes = base64.b64decode(response[i]['Vector'])
```

```
    vector = np.frombuffer(bytes, dtype=np.float64)

    # check if the vector is valid
    if np.any(np.isnan(vector)):
        print("Vector contains NaN values, skipping")
        continue

    all_vectors.append(vector)

    if len(all_vectors) == 0:
    print("No valid vectors found from input keywords")
    return None

    return np.mean(all_vectors, axis=0)
```

La funzione non esegue nessuna nuova operazione matematica interessante rispetto ad altre istanze di calcolo dei vettori, fatta eccezione di alcuni controlli per assicurarsi la validità della risposta.È importante sottolineare che, data la necessità di conversione dei dati da un linguaggio ad un altro (javascript/python), il vettore è stato convertito in un formato codificabile e decodificabile da entrambi: base64.
A questo punto ha inizio il ciclo di operazioni per calcolare lo score e generare i suggerimenti. Dopo aver ricevuto i documenti da una chiamata POST al server NodeJS (oppure dal server eXist, a seconda della modalità di utilizzo), ha inizio il calcolo dello score.

```
    # calculate affinity score
    if config.USE_EXIST:

        doc_keywords = document['Keywords'].split(',')

        for keyword in doc_keywords:
            if keyword in keywords:
                keyword_score += 1

        # get the score into a %
        keyword_score = keyword_score / len(doc_keywords) *
100
```

```
          keyword_score = round(keyword_score, 2)
     else:
          doc_keywords =
get_keywords_from_exist(document['FRBREXPRESSIONTHIS'],
database)

          if my_keywords_index is None or
np.any(np.isnan(document['DocIndex'])):
               current_suggestion['message'] = "Score not
calculated, there could be an issue with the document index
or the eurovocs in the db"
          else:
               # calculate the score with the cosine
similarity
               keyword_score = np.dot(my_keywords_index,
document['DocIndex']) / (np.linalg.norm(my_keywords_index) *
np.linalg.norm(document['DocIndex']))
               # get the score into a %
               if isinstance(keyword_score, np.ndarray):
                    print(keyword_score)
               keyword_score = round(keyword_score*100, 2)


     print(f"Keyword score: { keyword_score }")
     if keyword_score < config.MINIMUM_SCORE or
np.isnan(keyword_score):
          print("Score too low or not valid")
```

Il calcolo dello score che ci interessa è quello nella parte dell'else, dove viene assegnata alla variabile "keyword_score" il risultato.
Il calcolo del punteggio di similarità coseno può essere rappresentato come una formula matematica nel seguente modo:

$$[\text{keyword\_score} = \frac{\text{prodottoscalare}(\text{my\_keywords\_index}, \text{document}['\text{DocIndex}'])}{||\text{my\_keywords\_index}|| \times ||\text{document}['\text{DocIndex}']||}]$$

dove:

- $(\text{prodotto scalare}(\text{my\_keywords\_index}, \text{document}[\text{'DocIndex'}]))$ **indica il prodotto scalare tra il vettore che rappresenta le parole chiave di interesse** $((\text{my\_keywords\_index}))$ **e il vettore che rappresenta l'indice del documento** $((\text{document}[\text{'DocIndex'}]))$.
- $(\|\text{my\_keywords\_index}\|)$ **rappresenta la norma euclidea (magnitudine) del vettore** $(\text{my\_keywords\_index})$.
- $(\|\text{document}[\text{'DocIndex'}]\|)$ **rappresenta la norma euclidea (magnitudine) del vettore** $(\text{document}[\text{'DocIndex'}])$.

Infine, per esprimere il punteggio come percentuale, la formula moltiplica il punteggio risultante per 100 e lo arrotonda a due cifre decimali:

$$[\text{keyword\_score} = \text{round}\,(\text{keyword\_score} \times 100, 2)]$$

Dopo il calcolo del punteggio, viene dinamicamente generata la stringa del suggerimento che rispetta il formato AKN dei riferimenti, includendo un contatore per i riferimenti generati ed includendo tutti i qualificatori trovati in precedenza.

A questo punto il calcolo del punteggio e la generazione del suggerimento vengono ripetuti per ogni documento trovato, ed al termine di questo flusso di lavoro, eseguiamo uno step aggiuntivo prima di verificare se è stato generato un suggerimento utile da restituire. Vengono manipolate le informazioni ricavate nella seconda fase di questa esecuzione, nello specifico viene rimosso una singola cifra dal numero sequenziale e dall'anno trovati, così da permettere alle chiamate al database di tornare un insieme di documenti più variegato ma di mantenere coesione con la stringa in input (per esempio, il numero sequenziale 406 diventerebbe 40).

```
    # if the highest score is too low, we shall do more
loose queries to find a score that matches the keywords
    # let's remove a character from the info we have and try
again
    # there's not much more that can be done to find a
fitting document
    for info in list_of_info:
```

```
            # if the info are already 1 character long (or
missing), there's no point in removing more, remove them from
the list
            # this way we can avoid very big batches of
documents to check, making the component faster
            if (info['year'] is None or len(info['year']) == 1
or info['year'] == 20) and (info['sequential_number'] is None
or len(info['sequential_number']) == 1):
                # remove the info from the list
                list_of_info.remove(info)
                continue

            if info['year'] is not None and len(info['year']) >
1:
                info['year'] = info['year'][:-1]
            if info['sequential_number'] is not None and
len(info['sequential_number']) > 1:
                info['sequential_number'] =
info['sequential_number'][:-1]
            # ignoring doctype for now, assuming at least that
one is correct
```

Infine, L'applicativo effettua tre controlli chiave per decidere se effettuare una nuova iterazione di ricerca e generazione suggerimenti. I controlli sono:

```
max_score < config.MINIMUM_SCORE and workflows <
config.MAX_WORKFLOWS and len(list_of_info) > 0:
```

1. "max_score < config.MINIMUM_SCORE" , che verifica se lo score più alto trovato è minore di una soglia definita nella configurazione, che rappresenta il minor punteggio di un suggerimento per essere considerato accettabile;
2. "workflows < config.MAX_WORKFLOWS", che verifica se abbiamo superato il numero massimo di iterazioni di ricerca e generazione consentiti durante l'esecuzione.
3. "len(list_of_info) > 0", che si accerta che la lista di informazioni da cui ricercare i documenti non sia vuota.

Se almeno uno di questi requisiti non è soddisfatto, il programma ritorna i suggerimenti generati fino ad ora e termina. In caso contrario, effettua una nuova iterazione.

Capitolo 5 - Risultati

---

1. “Regulation 40”.
   Numero risultati: 10
   Numero risultati corrispondenti: 10
   Numero di risultati con score significativo: 10

2. “Regulation 406”
   Numero risultati: 10
   Numero risultati corrispondenti: 10
   Numero di risultati con score significativo: 10
3. “Regulation 4066”
   Numero risultati: 9
   Numero risultati corrispondenti: 0
   Numero di risultati con score significativo: 9
   Una query al DB MySQL è stata effettuata per individuare falsi negativi, di seguito richiesta e risultato:

```
'{
"DocNumber": 4066,
"DocType": "regulation"
}'
```

   Response body

```
[]
```

   non esiste un documento che corrisponde ai due parametri inseriti.
4. “Regulation EU 406”
   Numero risultati: 10
   Numero risultati corrispondenti: 10
   Numero di risultati con score significativo: 10
5. “Regulation (EU) 406”
   Numero risultati: 10
   Numero risultati corrispondenti: 10
   Numero di risultati con score significativo: 10
6. “Regulation 407”
   Numero risultati: 9
   Numero risultati corrispondenti: 9
   Numero di risultati con score significativo: 9
7. “Regulation 1233”
   Numero risultati: 7
   Numero risultati corrispondenti: 7
   Numero di risultati con score significativo: 7
8. “Regulation 233”
   Numero risultati: 7

Numero risultati corrispondenti: 7

Numero di risultati con score significativo: 7

9. “Directive 1024”

Numero risultati: 1

Numero risultati corrispondenti: 1

Numero di risultati con score significativo: 1

10. “Directive 1024”

Numero risultati: 1

Numero risultati corrispondenti: 1

Numero di risultati con score significativo: 1

11. “Directive 2019/1024”

Numero risultati: 1

Numero risultati corrispondenti: 0

Numero di risultati con score significativo: 1

12. “Directive (EU) 1024”

Numero risultati: 1

Numero risultati corrispondenti: 1

Numero di risultati con score significativo: 1

13. “Directive (EU) 1024”

Numero risultati: 1

Numero risultati corrispondenti: 1

Numero di risultati con score significativo: 1

14. “Directive (EU) 2019/1024”

Numero risultati: 1

Numero risultati corrispondenti: 0

Numero di risultati con score significativo: 1

15. “Decision 406”

Numero risultati: 2

Numero risultati corrispondenti: 2

Numero di risultati con score significativo: 2

16. “Decision 406/2014”

Numero risultati: 1

Numero risultati corrispondenti: 0

Numero di risultati con score significativo: 1

17. “Decision 406/2014/EU”

Numero risultati: 1

Numero risultati corrispondenti: 0

Numero di risultati con score significativo: 1

18. “406/2014/EU”

    Numero risultati: 1

    Numero risultati corrispondenti: 1

    Numero di risultati con score significativo: 1

19. “406/EU”

    Numero risultati: 12

    Numero risultati corrispondenti: 12

    Numero di risultati con score significativo: 12

20. “406”

    Numero risultati: 12

    Numero risultati corrispondenti: 12

    Numero di risultati con score significativo: 12

Capitolo 6 - Conclusioni e sviluppi futuri

Il componente facilita i prossimi passi nello sviluppo di nuovi componenti grazie alle solide basi che pone tramite il server NodeJS ed il database MySQL. Questo è uno dei primi motivi per cui sono davvero soddisfatto del lavoro svolto finora. Ho onorato il progetto a cui ho preso parte rispettando i principi posti all'inizio dello sviluppo, come i tempi di risposta brevi, una facile accessibilità ed una solida documentazione per utenti e futuri sviluppatori.
Sicuramente uno dei punti di vista più interessanti del progetto è l'utilizzo di strumenti AI affiancati da meccanismi di validazione dei risultati. Al momento, gli strumenti più diffusi di intelligenza artificiale non sono in grado di verificare accuratamente tutte le informazioni che provvedono nella loro risposta. Nel caso di questo componente, tuttavia, ogni risultato parte da una base di dati affidabile con informazioni ricavate da fonti ufficiali provenienti dall'European Publication Office. Questo assicura delle risposte significative e prive di informazioni errate riguardo i documenti citati.
Nonostante la funzionalità del componente, ci sono alcuni aspetti che concedono un margine di miglioramento. La più importante è sicuramente integrare nuove funzionalità sulla base di una LLM al momento in fase di lavorazione in sede. L'utilizzo di un modello linguistico allenato sulla base degli stessi documenti che dovremmo assistere alla scrittura apre moltissime possibilità per sperimentare con nuove feature di questo ed altri futuri componenti.
Successivamente, è doveroso aggiungere ulteriori controlli di validità ai suggerimenti in risposta. In particolare, la validazione tramite qualificatori e data di validità rientra nei prossimi piani di aggiornamento del componente. La possibilità di segnalare all'utente eventuali errori ed inconsistenze presenti anche nella citazione del singolo articolo, paragrafo, lista, punto del riferimento sarà sicuramente utile se non necessaria ad un suggeritore per citazioni. Con dei controlli aggiuntivi sulle date di validità dei documenti presenti viene aggiunta la possibilità di segnalare all'utente eventuali decisioni, direttive o normative attualmente abrogate.
Il modo in cui i riferimenti vengono scritti è cambiato nel tempo, e perciò un autore ha bisogno di conoscere il formato in utilizzo del periodo in cui la normativa o direttiva di riferimento è stata scritta. Perciò, ancora una volta venendo in aiuto con questo progetto, il componente potrà beneficiare da un algoritmo che rilevi il formato di scrittura del documento in base alla data di rilascio, e ritorni nel suggerimento il formato corretto per la citazione.

Bibliografia

[GPF12] M.P. Giovannini, Monica Palmirani, E. Francesconi, *LINEE GUIDA PER LA MARCATURA DEI DOCUMENTI NORMATIVI SECONDO GLI STANDARD NORMEINRETE*, FIRENZE, European Press Academic Publishing, 2012, pp. 200 (2012).

[PA12] Monica Palmirani, *Legislative XML: Principles and technical tools*, ROMA, Aracne, 2012 (2012).

[PA19] Monica Palmirani, "Akoma Ntoso for Making FAO Resolutions Accessible" in: *Knowledge of the Law in the Big Data Age*, Amsterdam, IOS Press, *FRONTIERS IN ARTIFICIAL INTELLIGENCE AND APPLICATIONS*, 2019, 317, pp. 159–169 (2019).

[PA22] Monica Palmirani, "A Smart Legal Order for the Digital Era: A Hybrid AI and Dialogic Model," *RAGION PRATICA*, 2-2022, pp. 633–65 (2022).

[PS21] Monica Palmirani, Francesco Sovrano, Davide Liga, Salvatore Sapienza, Fabio Vitali, "Hybrid AI Framework for Legal Analysis of the EU Legislation Corrigenda," in: *Legal Knowledge and Information Systems*, *FRONTIERS IN ARTIFICIAL INTELLIGENCE AND APPLICATIONS,* 2021, pp. 68–75 (2021).

[PV11] Monica Palmirani, Fabio Vitali, "Akoma-Ntoso for Legal Documents," in: *Legislative XML for the Semantic Web. Principles, Models, Standards for Document Management*, BERLIN, Springer Verlag, 2011, pp. 75–100 (2011).

[PV12] Monica Palmirani, Fabio Vitali, "Legislative Drafting Systems," in: *Usability in Government Systems*, NEW YORK, Morgan Kaufmann, 2012, pp. 133–151 (2012).

[PV14] Monica Palmirani, Fabio Vitali, Albano Bernasconi, Luca Gambazzi, "Swiss Federal Publication Workflow with Akoma Ntoso," in: *Legal Knowledge and Information Systems*, Amsterdam, IOS Press, 2014, 271, pp. 179–184 (2014).

[PV18] Akoma Ntoso Version 1.0. Part 1: XML Vocabulary (2018).

[PV22] Monica Palmirani, Fabio Vitali, "Drafting legislation in the era of AI and digitisation" (2022) https://joinup.ec.europa.eu/sites/default/files/document/2022-06/Drafting%20legislation%20in%20the%20era%20of%20AI%20and%20digitisation%20%E2%80%93%20study.pdf.

[TMG+17] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, Armand Joulin, "Advances in Pre-Training Distributed Word Representations," arXiv preprint arXiv:1712.09405 (2017).

Indice Immagini

Ringraziamenti

Ringrazio la prof.ssa Monica Palmirani per aver supervisionato la stesura di questa dissertazione e per la magnifica opportunità concessa nella sede CIRSFID-ALMA-AI.
Ringrazio i miei genitori, Luciano e Maria Teresa, per la possibilità di formazione che mi è stata concessa qui all'Unibo, che mi ha fatto crescere da un punto di vista professionale e personale, in moltissimi aspetti.
Ringrazio tutti i miei amici, vecchi e nuovi, per avermi sopportato finora.