MOTHER 2022

Autore:
LONGO ANGELO 761056
NATALE GIUSY 766401

Progetto Mother

Obiettivo: Sviluppo di un sistema IoT per il monitoraggio della salute materno-fetale mediante l'utilizzo di un sistema di sensori wearable.
Tracciare la crescita del feto è essenziale per la salute materna e aiuta a prevenire la mortalità alla nascita.

I sensori vengono gestiti da una scheda Arduino Sense e prevedono:
• Misura;
• Posizione data da una IMU a nove Gradi di Libertà.

L'IMU integra:
• un accelerometro in grado di fornire una posizione relativa della madre dopo uno spostamento (soggetto ad errori di deriva);
• una bussola per fornire l'angolo;
• un barometro per la pressione atmosferica;

Ulteriori sensori utilizzati per la raccolta dati sono un sensore di battito cardiaco e uno per l'ossigenazione del sangue (fotoplestimografia).
Fondendo questi dati, attraverso un elettrocardiogramma (battito cardiaco – dato strutturato) e una fotoplestimografia (ossigenazione – dato strutturato) possiamo ricavare utili informazioni sulla pressione sanguigna.

I dispositivi utilizzati nel sistema prevedono:
• dispositivo wearable per la raccolta dati (Arduino); EDGE
• dispositivo smartphone + pc per la consultazione dei dati; FOG
• infrastruttura online per il calcolo; CLOUD

Il sistema prevede di poter catturare il parlato (speech to text) della madre per poi convertirlo in testo ed essere utilizzato come supporto ad una consulenza medica.

Nello sviluppo della base di dati la situazione diventa complicata, a causa del regolamento sulla protezione dei dati personali.

Per il GDPR esiste il concetto di *Data vault*, ovvero un contenitore locale il cui controllo è esclusivo del paziente.
Questo contenitore raccoglie tutti i dati e le rilevazioni del singolo paziente.
Possiamo immaginare il *Data vault* come una porzione del database avente il dispositivo, i sensori e le rilevazioni. Ogni sensore effettua una rilevazione, memorizzata in un file o in una tabella di un database da interrogare al bisogno. Questo è un database a disposizione di ciascuna mamma.

Dall'altra parte abbiamo il medico che ha l'accesso ad un database contenente le mamme e le patologie ad esse associate.
Il medico non può accedere senza consenso esplicito ai dati che sono stati acquisiti dalla mamma.
Il trattamento dei dati sanitari è gestito e tutelato dal regolamento GDPR.

Il Personal Data Vault permette di memorizzare le informazioni relative ai controlli automatici effettuati attraverso i sensori e i referti di visite ed esami effettuati di persona.

Gli studi clinici che possono essere condotti sull'analisi di questi dati sono di due tipi:

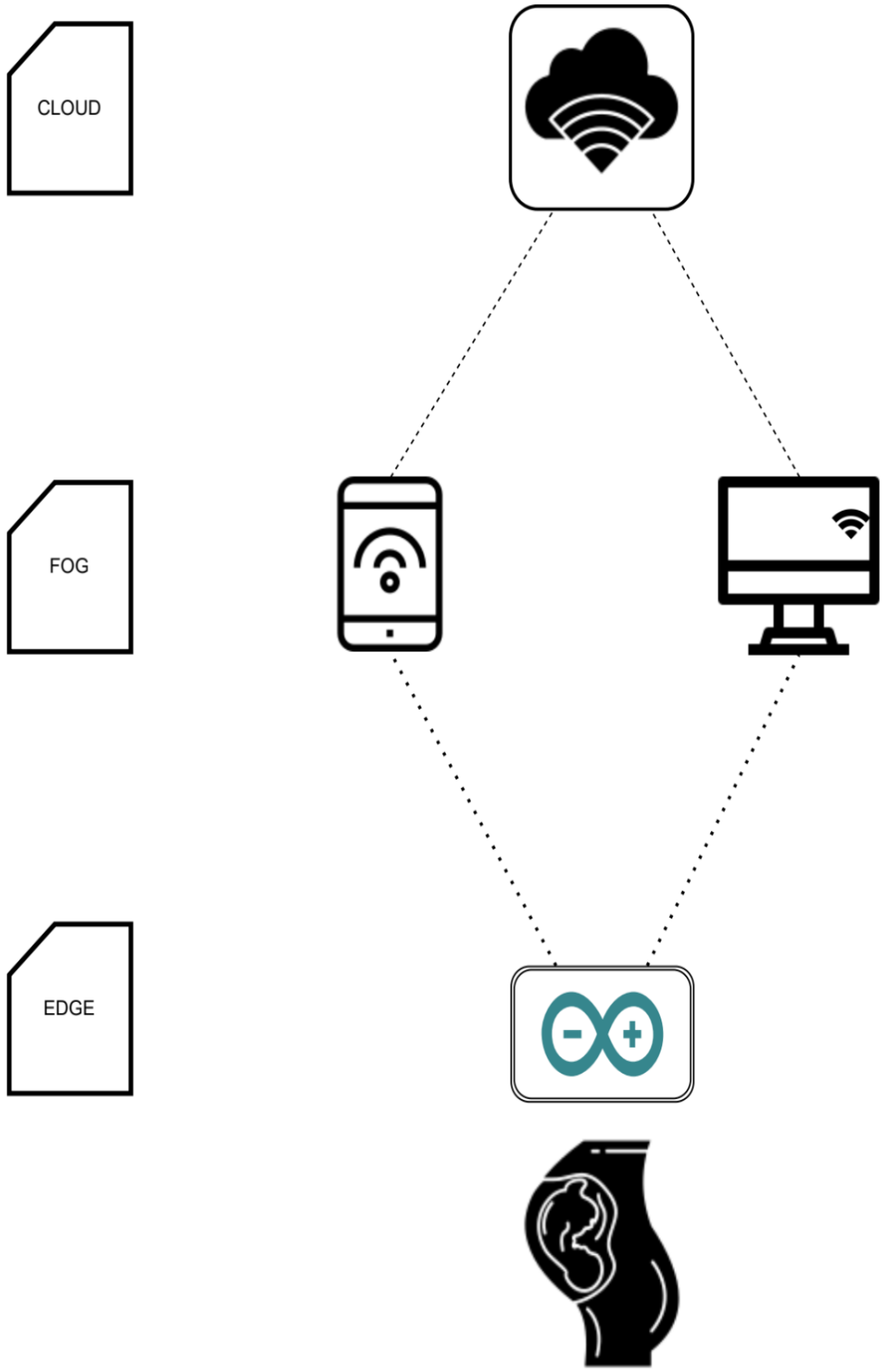
- Prospettici: **valutano gli effetti di un intervento seguendo le persone coinvolte a partire dall'inizio dello studio e fino alla sua conclusione, per osservare gli esiti dell'intervento stesso.**
- Retrospettivi: **misurano eventi accaduti in un periodo precedente rispetto al disegno dello studio.**

Abbiamo deciso di analizzare uno scenario possibile di questo sistema.
Una mamma possiede un dispositivo munito di due sensori, con il quale vengono rilevati dati:

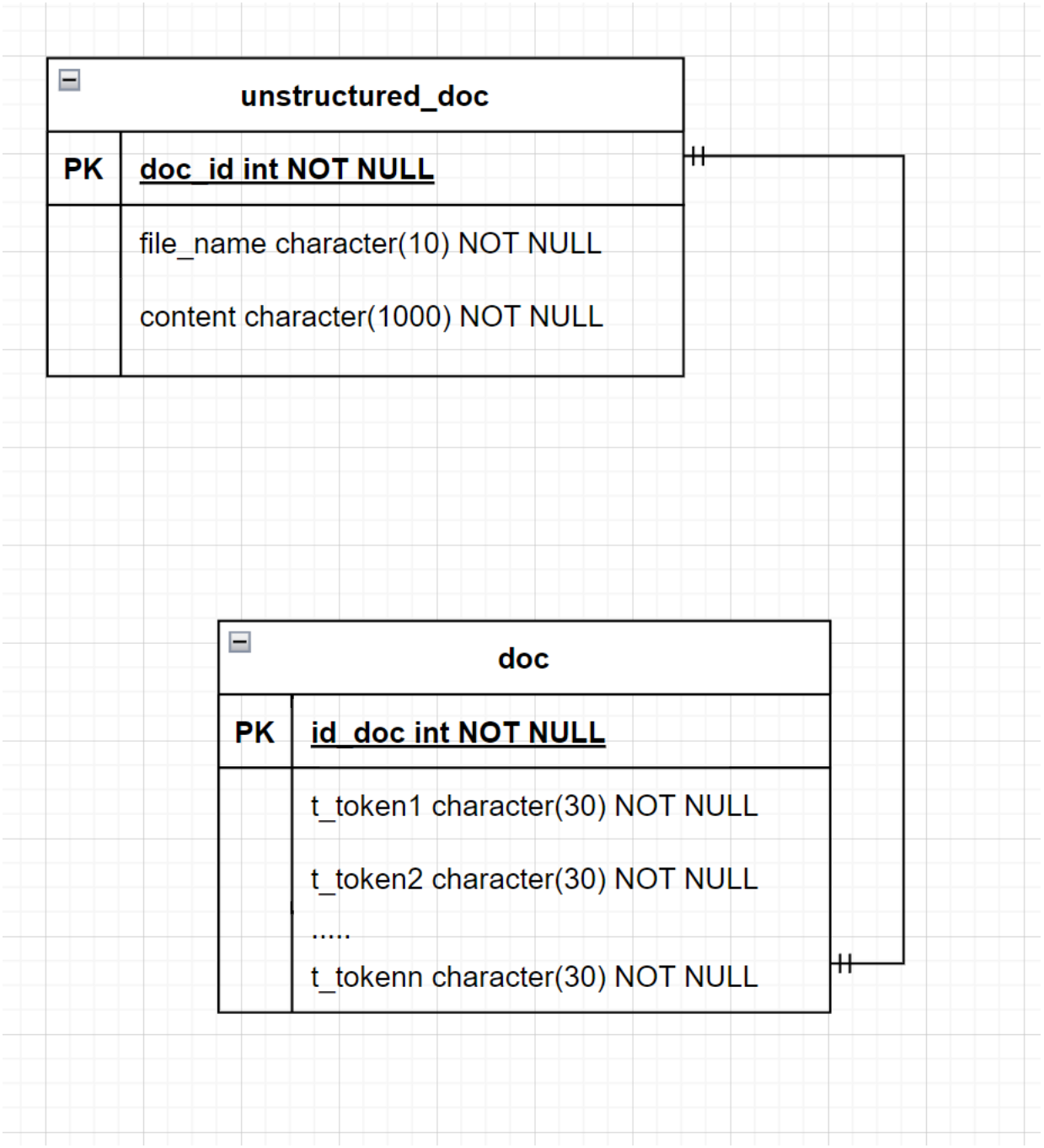
- sull'accelerazione dei battiti del feto;
- sulle decelerazioni di varia entità;
- sulle contrazioni uterine;

Il medico, dopo aver ottenuto il consenso dalla paziente ha accesso a questi dati.

Struttura del sistema



Schemi entità-relazioni



File Python: NLP_V8

Il DBMS scelto per la creazione del database è PostgresSQL.
I documenti analizzati sono 55 e rappresentano conversazioni mediche e commenti su alcune patologie relative alla gravidanza.
Inizialmente l'obiettivo di questa parte di progetto era quello di strutturare un database che permettesse di catalogare le patologie analizzate nei documenti testuali con le relative cure.
Procedendo con l'analisi abbiamo notato che i documenti non riportano in maniera precisa le tematiche e non è possibile ricavare le due entità ai fini della strutturazione.

Abbiamo proceduto quindi alla creazione di una pipeline NLP Natural Language Processing in linguaggio **Python**, optando fin da subito alla libreria SpaCy.

L'elaborazione del linguaggio naturale comporta una successione di fasi che tentano di superare le ambiguità del linguaggio umano, si tratta di un processo particolarmente delicato a causa delle complesse caratteristiche del linguaggio stesso. Proprio per ridurre il più possibile gli errori, il processo di elaborazione viene suddiviso in numerose fasi di seguito esplicitate.

NLTK e SpaCy sono due delle più popolari librerie disponibili in Python. Entrambe possono teoricamente svolgere tutti i task presenti in NLP ma ci sono differenze sostanziali che hanno portato alla scelta di una di esse.

NLTK è nato come strumento in ambito scolastico e di ricerca e permette di scegliere tra una vasta gamma di algoritmi per l'analisi del linguaggio.
È essenzialmente una libreria per l'elaborazione delle stringhe, dove ciascuna funzione prende in input una stringa e la fornisce in output processata.

SpaCy invece è una libreria object-oriented dove ogni funzione ritorna un oggetto, molto più facile da manipolare rispetto a stringhe e vettori.
L'utilizzo degli oggetti permette agli sviluppatori di non dover necessariamente controllare la documentazione per capire il funzionamento del sistema.

Abbiamo optato per quest'ultima sia per la gestione semplificata degli oggetti e per la velocità di esecuzione dei comandi.

Sotto il punto di vista delle performance e della gestione della memoria, SpaCy è la scelta ottimale.

Tra la vasta gamma di modelli proposti da SpaCy abbiamo puntato sull'efficienza. Una pipeline veloce e di dimensione piccola a discapito dell'accuratezza.
Nel nostro caso l'analisi da condurre sui documenti non andrà oltre il riconoscimento dell'entità e di tag, ci è sembrato quindi più conveniente puntare sulle prestazioni nell'esecuzione.

"en_core_web_sm" → modello scelto per l'analisi

Il file python è stato fin da subito impostato per consentire una lettura dinamica dei documenti all'interno della cartella /docs/.
Ciascun documento è chiamato con doci.txt dove i varia da 1 a 55 (il numero totale dei documenti presenti).
Per consentire l'analisi dell'intero corpus è stata creata una variabile che contenesse in maniera sequenziale tutti i documenti separati da uno spazio.

Utilizzando poi la funzione *displacy* abbiamo condotto una prima osservazione per capire il tipo di entità coinvolte nel corpus.
Essendoci molti numeri dalla visualizzazione notiamo subito una presenza preponderante di *ORDINAL* per i numeri ordinali, *CARDINAL* per i numeri cardinali e *DATE* per le date.

Le prime fasi condotte nell'elaborazione del testo riguardano il riconoscimento di token. L'elaborazione di un testo inizia con la sua scomposizione in token corrispondenti a spazi, parole, punteggiatura, frasi. Il task non è particolarmente complesso rispetto agli altri, ma presenta comunque alcune problematiche: per esempio, se si considera il punto come fine di una frase, si rischia di sbagliare frequentemente in quanto il punto potrebbe riferirsi a una abbreviazione, a una data o a un link.

Il numero totale di token analizzati dopo il Sentence splitting e Tokenization è pari a 3730.
Per la creazione del dizionario da utilizzare per le successive analisi è stato fatto un POS Tagging (Part of Speech Tagging) per ciascuna sentence presente nel corpus.

La morfologia fornisce informazioni sulla forma di ciascuna parola e sul suo ruolo all'interno di una frase. Il lessico (o vocabolario) è il complesso delle parole e dei modi di dire (locuzioni) di una lingua.

L'analisi morfologica e lessicale prevede la consultazione di apposite liste di lemmi e delle loro derivazioni (liste che possono essere eventualmente integrate con termini specifici relativi al dominio che si sta studiando), la risoluzione delle forme di flessione (come la coniugazione per i verbi e la declinazione per i nomi) e la classificazione delle parole basata su determinate categorie (come il nome, il pronome, il verbo, l'aggettivo).

Analizzatori lessicali e morfologici possono essere implementati separatamente, ma spesso costituiscono un unico task.

In questa fase abbiamo filtrato i token in base agli stop_word (2149 dei 3730 token, più del 50%), alla punteggiatura e ai nomi.
Il dizionario creato comprende 818 token (383 non ripetuti).

Il modello di rappresentazione testuale scelto è il modello Vettoriale, che ci permette di esplicitare l'importanza di key-words in un doc.

L'importanza viene stabilita associando a ciascun termine un peso locale (tf), un peso globale (idf) e una costante di normalizzazione, utilizzata in presenza di documenti voluminosi.
La libreria utilizzata per la creazione della matrice termini documenti è la TfidfVectorizer di sklearn.

L'utilizzo di questo pacchetto ci consente di fornire in input un dizionario di key-words e di generare una matrice dove, per ciascun documento, compaiono i termini presenti in tutto il documento, correlati dal peso locale.

Ogni doc utilizza quindi una porzione dell'intero dizionario per essere rappresentato rispetto all'intero corpus. Questa struttura è chiamata surrogato.

Il surrogato sintetizza il contenuto del dato non strutturato (potrebbe non rappresentarlo completamente) e permette di individuare uno schema logico rispetto all'uso.

Per la strutturazione dei surrogati abbiamo creato due versioni differenti.

La prima utilizza tutti i termini che hanno un tf-idf positivo, quindi effettivamente presenti all'interno del documento:

```
{'Doc1': [('medications', 1.6931471805599454),
  ('vitamins', 1.0),
  ('minerals', 1.0),
  ('supplements', 1.0),
  ('pregnancy', 1.0),
  ('ob', 1.0),
  ('gyn', 1.0),
  ('prescription', 1.0),
  ('counter', 1.0)],
 'Doc2': [('pregnancy', 1.0),
  ('ob', 1.0),
  ('testing', 1.6931471805599454),
  ('way', 1.0),
  ('baby', 1.0),
  ('abnormalities', 1.6931471805599454),
  ('tube', 1.0),
  ('defects', 1.0),
  ('chances', 1.0),
  ('percent', 1.0),
```

La seconda invece è generata creando un filtro basato sull'idf di ciascun termine. L'Inverse Document Frequency è il peso globale che quel token ha rispetto a tutto il corpus. Rimuovendo i token con gli idf relativamente più bassi, andiamo a ridurre sicuramente la dimensione del dizionario preso in considerazione, e riduciamo anche la sovra importanza rispetto all'intero corpus.

Il dizionario ridotto comprende 375 token.

```
for ele1, ele2 in zip(tfidfvectorizer.get_feature_names(), tfidfvectorizer.idf_):
    if 2.7<ele2<4.5:
        count_3 += 1
```

L'obiettivo iniziale del progetto era quello di strutturare un database che comprendesse le patologie e le cure rilevate in fase di gravidanza.
Non essendo realizzabile senza l'utilizzo approfondito di tecniche di text mining, abbiamo optato per la strutturazione matriciale dei termini con i diversi documenti.

Sono state create due tabelle:

- la prima raccoglie i documenti nel loro formato non strutturato
- la seconda ingloba i diversi surrogati dei documenti

La prima tabella consente di ottenere il documento vero e proprio in fase di query-matching con le key-words.

La seconda tabella comprende per ciascun documento i vari termini presenti nel surrogato, inseriti con un 't_' all'inizio, per evitare problemi di conflitto con l'ambiente sql.

Per sfruttare in qualche modo l'informazione relativa alle patologie e alle cure nelle query, abbiamo utilizzato un ulteriore libreria chiama ScispaCy.

ScispaCy è una libreria software open-source per l'elaborazione avanzata del linguaggio naturale, scritta nei linguaggi di programmazione Python e Cython. La libreria offre modelli di reti neurali statistiche per l'elaborazione di testi biomedici, scientifici o clinici. Con scispaCy è possibile costruire facilmente modelli statistiche linguisticamente sofisticati per una varietà di problemi di NLP.

Sfruttando le potenzialità di questo pacchetto, abbiamo ri-analizzato il corpus per trovare tutti i token corrispondenti all'ambito biomedico.
Il riconoscimento delle entità ha permesso poi di generare un vettore contenente tutte le patologie presenti e un vettore per le cure ad esse relative.

Questi vettori sono stati utilizzati successivamente per filtrare l'intero dizionario e trovare i termini corrispondenti con i surrogati creati.

Questo ha permesso di generare 32 query, suddivise per patologie e cure, permettendo così di raffinare la fase di query-matching pur avendo un dataset altamente sparso.

Sono stati infine generati i 3 file sql (ddl,dml,ql), direttamente lanciabili in PostgreSQL.