Archivist Agent

Progetto Ingegneria della Conoscenza AA. 2020/2021

Gruppo formato da:

Donato Lucente mat. 622810 d.lucente2@studenti.uniba.it

Link repository: https://github.com/donakunn/Archivist-Agent

Indice

1. Introduzione

Archivist Agent è un sistema che simula il comportamento di un robot all'interno di un archivio di articoli di giornali, in grado di compiere operazioni quali la classificazione degli articoli di giornale per genere e l'archiviazione del suddetto articolo nell'archivio corrispondente al genere predetto. L'archivio è inoltre fornito di una serie di sensori preposti alla localizzazione del robot durante i vari spostamenti.

1.1 Informazioni utili

Il codice è suddiviso in package, in particolare: "Classifiers", package contenente le classi che implementano il classificatore usato dal sistema, "Search", contenente le classi che implementano la ricerca, "Localization", contenente le classi che implementano la localizzazione del robot all'interno dell'archivio. Nella directory principale è presente lo script "main.py" da utilizzare per avviare il sistema.

È possibile scaricare il dataset all'indirizzo: http://qwone.com/~jason/20Newsgroups/

Nello specifico è stata utilizzata la versione "20news-bydate", contenente 20 cartelle, una per genere, ognuna contenente centinaia di articoli.

Per implementare all'interno del sistema la ricerca Breach and bound e Hidden Markov Model con particle sampling, sono stati utilizzati gli algoritmi forniti con il libro di testo.

Per testare il sistema sono stati inclusi nella cartella del progetto tre articoli di giornale in formato txt, ovvero "newsAuto.txt", "newsSpace.txt", "newsIBM.txt" .

Il sistema viene distribuito con il classificatore già addestrato. In caso si voglia riaddestrare il sistema occorre eliminare i file "prob_classi.npy", "prob_parole.npy" e "vocabolario.npy".

2. Classificatore

Per l'implementazione del sistema è stato implementato un classificatore di tipo naive Bayes dove la fase di training consiste nel:

- Costruire il corpus dei documenti dal dataset, il particolare il corpus viene strutturato in dizionario con chiave Genere e valore: dizionario con chiave documento e valore lista di parole contenute all'interno del documento.

NB. Per migliorare le prestazioni del sistema questo dizionario è stato implementato ignorando eventuali stopwords, data la loro alta frequenza all'interno dei documenti e il loro scarso potere informativo.

- Costruire il vocabolario equivalente, usando il modello bag of words, come dizionario parola-frequenza parola all'interno del dataset. Per ragioni di efficienza sono state rimosse dal dizionario tutte le parole con frequenza minore di 3, così da evitare di calcolare probabilità infinitesimali, il cui contributo alla classificazione

sarebbe stato quasi inesistente.

- Costruire il dizionario contenente la probabilità delle varie classi C_i calcolate come rapporto tra numero di documenti di classe C_i nel training set e numero di documenti totali all'interno del training set.
- Costruire il dizionario contenente per ogni categoria, la probabilità condizionata della j-esima parola data la categoria, calcolata come il rapporto tra il numero di volte in cui la parola in questione compare nei documenti di classe C_i e il numero totale di parole nella classe C_i.

NB. Al calcolo delle probabilità condizionate è stata applicata la correzione di Laplace, sommando uno a numeratore e la cardinalità del vocabolario a denominatore, così da non avere probabilità nulle, inoltre, per evitare errori di underflow, dovuti alla moltiplicazione di probabilità molto piccole, viene calcolato il logaritmo naturale della probabilità così da poter trasformare il prodotto in una somma di logaritmi.

Finita la fase di training, per attribuire a un documento la classe più probabile, vengono calcolate le probabilità condizionate: $P(d \mid C_i)$ per ogni categoria, utilizzando il teorema di Bayes, con l'assunzione di indipendenza delle varie parole (rendendolo di fatto un classificatore naive Bayes) e tralasciando il calcolo del denominatore, irrilevante per il confronto tra probabilità. Dopodiché viene assegnata al documento da classificare la categoria con la probabilità più elevata.

NB per evitare di dover ri-addestrare il classificatore ogni volta che si avvia il sistema, dopo la fase di training i dizionari vengono salvati su disco così da poterli riutilizzare al successivo avvio del sistema.

2.1 Test e risultati

Finito l'addestramento il sistema viene testato un corpus di documenti diversi da quello di training, con i seguenti risultati:

Classe alt.atheism 83.07% di risposte esatte

Classe comp.graphics 79.95% di risposte esatte

Classe comp.os.ms-windows.misc 40.10% di risposte esatte

Classe comp.sys.ibm.pc.hardware 79.08% di risposte esatte

Classe comp.sys.mac.hardware 84.94% di risposte esatte

Classe comp.windows.x 77.47% di risposte esatte

Classe misc.forsale 80.00% di risposte esatte

Classe rec.autos 91.67% di risposte esatte

Classe rec.motorcycles 95.23% di risposte esatte

Classe rec.sport.baseball 94.46% di risposte esatte

Classe rec.sport.hockey 96.49% di risposte esatte

Classe sci.crypt 91.92% di risposte esatte

Classe sci.electronics 73.79% di risposte esatte

Classe sci.med 82.32% di risposte esatte

Classe sci.space 90.36% di risposte esatte

Classe soc.religion.christian 93.47% di risposte esatte

Classe talk.politics.guns 91.48% di risposte esatte

Classe talk.politics.mideast 87.77% di risposte esatte

Classe talk.politics.misc 61.61% di risposte esatte

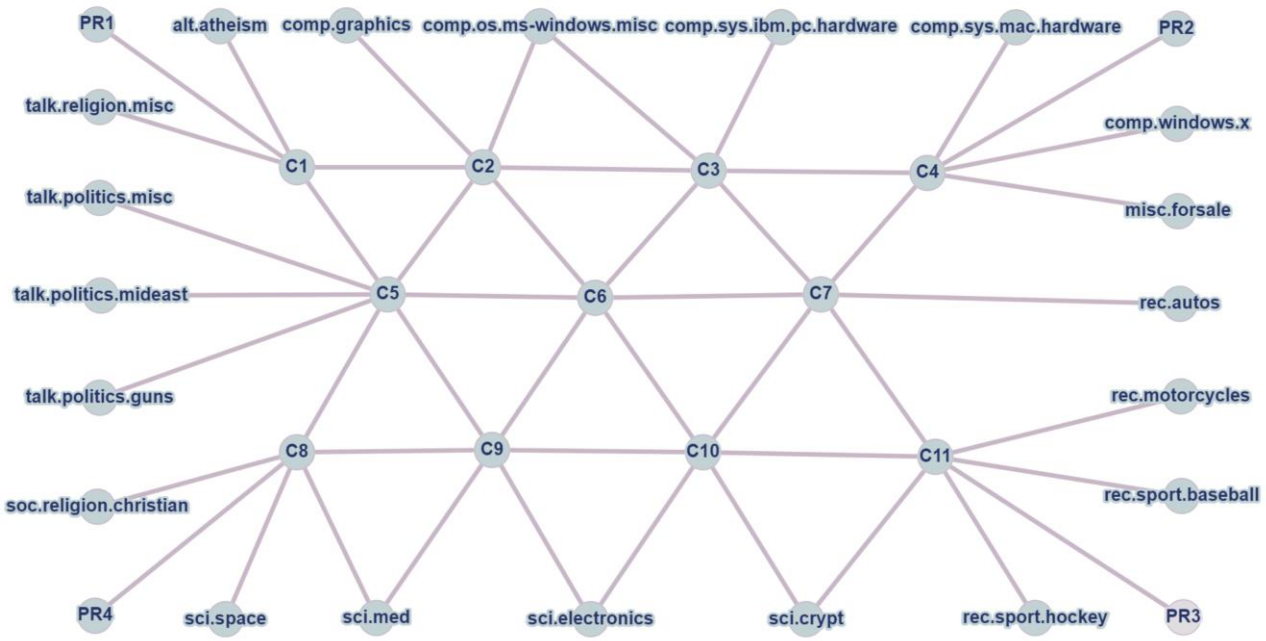
Classe talk.religion.misc 55.38% di risposte esatte

Risultato totale della classificazione: 82.21% di risposte esatte

3 Ricerca percorso

Una volta addestrato il classificatore è possibile assegnare al robot un documento da archiviare, per farlo occorre specificare il nome del documento desiderato, senza includere l'estensione del file (ad esempio ".txt").
A quel punto il robot lo classifica con la categoria più probabile, calcola il percorso migliore verso la posizione dell'archivio corrispondente, lo deposita, calcola la posizione verso il resting point più vicino (posizione di riposo per il robot, dove può essere ricaricato e/o ricevere un nuovo documento da archiviare) e si dirige in quella posizione, in attesa di nuovi ordini.
NB per questioni di semplicità, il documento deve essere in formato txt ed essere situato all'interno della cartella "Main" del progetto.
L'archiviazione viene simulata copiando il documento in una cartella corrispondente all'archivio individuato dal robot, all'interno della cartella del progetto.

3.1 Struttura dell'archivio



Per semplificare il problema l'archivio è stato concettualizzato come un grafo. I quattro resting point sono situati ai quattro angoli dell'archivio; all'avvio del sistema il robot si trova in uno dei quattro resting point scelto in maniera casuale. I nodi da C1 a C11 rappresentano stanze e corridoi vari, infine per ogni categoria di articolo di giornale è presente un nodo che rappresenta l'archivio corrispondente.

3.2 Algoritmo e funzionamento

Una volta classificato il documento, il sistema costruisce il grafo relativo all'archivio, istanziando i vari nodi e i vari archi pesati, dove il peso indica il tempo di percorrenza da un nodo ad un altro, tenendo in considerazione che archi tra nodi simili (da un resting point ad un nodo C ad esempio), hanno un tempo di percorrenza simile ma quasi sempre diverso.
La funzione euristica viene definita come una sottostima del tempo di percorrenza da un nodo ad un altro (ottenuta rimuovendo dal tempo di percorrenza tra un nodo di un certo tipo ad un altro, del tempo necessario a percorrere curve o aggirare ostacoli); In particolare, ogni volta che il robot debba spostarsi all'interno dell'archivio costruisce, sfruttando la programmazione dinamica, un dizionario contenente i valori della funzione euristica del nodo target verso ogni altro nodo del grafo, usando l'algoritmo di ricerca in ampiezza.
NB è stata scelta una sottostima del tempo come euristica per renderla ammissibile. È stato possibile sfruttare la programmazione dinamica con algoritmo di ricerca in ampiezza per il calcolo dell'euristica grazie al numero finito e limitato di nodi all'interno del grafo.

Per calcolare il percorso viene utilizzato l'algoritmo branch and bound, che unisce le euristiche calcolate in precedenza con l'algoritmo depth- first per ricercare il percorso ottimale, o percorso

più veloce verso il nodo target. Inizialmente il bound viene settato come una sovrastima del percorso più lungo tra un nodo ed un altro, in modo da considerare anche il nodo alla posizione opposta del grafo, evitando quindi di potare un percorso simile, ma non considerare percorsi più lunghi. A tale scopo viene utilizzato anche il cycle-pruning, che data la struttura del grafo, evita che il robot consideri tutti quei percorsi in cui vi è un nodo ripetuto, migliorando sensibilmente le performance dell'algoritmo.

NB La scelta dell'algoritmo di ricerca è ricaduta sull'algoritmo branch and bound in quanto oltre ad essere molto efficiente risulta particolarmente adatto a grafi dove vi sono più percorsi verso un nodo obbiettivo, come accade nel grafo implementato.

Una volta calcolato il percorso, il robot lo percorre e deposita l'articolo. Una volta terminata questa operazione, utilizza la funzione euristica per stimare la distanza di ogni resting point dalla sua posizione, sceglie quello con la distanza minima, calcola il percorso verso quel nodo riutilizzando l'algoritmo, lo percorre, dopodiché resta in attesa di un nuovo documento da depositare.

4. Localizzatore

Sebbene il sistema mostri il percorso calcolato dal robot, allo scopo di mostrare il corretto funzionamento dell'algoritmo di ricerca Breach and bound, l'utente non dovrebbe essere in grado di conoscere direttamente la posizione del robot, ma è possibile avere una predizione sulla posizione corrente, non rilevata con certezza ma con una certa probabilità, attraverso la simulazione di una serie di sensori, uno per stato del grafo, utilizzati per effettuare osservazioni sullo stato del sistema ed incorporarle alla definizione di un modello nascosto di Markov per calcolare le distribuzioni di probabilità dei vari stati, con lo scopo di rappresentare la probabilità che il robot si trovi nel relativo stato.

All'atto pratico, il sistema dopo aver mostrato il percorso calcolato verso la destinazione, mostra la posizione corrente del robot, lo fa avanzare di una posizione alla volta, mostra la posizione più probabile calcolata allo stage attuale e resta in attesa dell'input dell'utente per avanzare alla posizione successiva.

4.1 Implementazione del modello nascosto di Markov

Il modello in questione è implementato utilizzando i seguenti parametri:

- Lo stato del sistema allo stage corrente è rappresentato dall'insieme delle variabili, una per stato del grafo, atte a rappresentare la probabilità relativa al suddetto stato.
- Viene simulata la presenza di un sensore, uno per stato del grafo, con il compito di fornire una osservazione su tale stato.
- Viene definito un dizionario dove, per ogni sensore, inserito come chiave, viene inserito come valore un dizionario contenente la probabilità di individuare il robot da suddetto sensore, seguendo questa logica:
    - La probabilità del sensore di uno stato di rilevare il robot sullo stato stesso è posta pari al 95%
    - Dato il nodo corrente n, è possibile che il robot sia individuato da un qualsiasi altro sensore posto in un nodo m collegato da un arco al nodo iniziale n con probabilità del 5%
- Viene definito un dizionario dove, i vari stati vengono inseriti come chiave e per valore un dizionario contenente le probabilità di transizione da uno stato ad un altro come segue:
    - Se il robot si trova in uno stato del tipo 'resting point' resta su questo stato con probabilità del 10%, mentre si sposta su un nodo collegato con probabilità del 90%
    - Se il robot si trova in uno qualsiasi degli altri stati resta fermo con probabilità del 5%, altrimenti si sposta su un altro stato con probabilità del 95% fratto il numero di archi collegati al nodo in questione.
- All'avvio del sistema ogni nodo ha la medesima probabilità, pari a 1 fratto numero degli stati.

NB Per ragionare sullo stato corrente del sistema, la scelta è ricaduta sul modello nascosto di markov in quanto risulta un modello perfetto per un sistema che si evolve nel tempo, dove è possibile definendo delle probabilità a priori generare una belief network potenzialmente infinita e dove ogni stato dipende solo dallo stato immediatamente precedente, come nel caso di un robot che si sposta lungo un percorso fatto di stati.

È importante notare che, l'osservazione sul sistema è fatta specificando una serie di 1 e 0, dove 1 rappresenta il rilevamento del robot da parte di un sensore e 0 l'evento opposto. Inoltre, è possibile che un sensore che non abbia rilevato il robot non comunichi l'osservazione, a causa di un guasto o di un problema di natura simile.

Questo comportamento occorre con probabilità del 5%.

Per il calcolo delle distribuzioni relative ai vari stati, incorporando anche le varie osservazioni date dai sensori, viene utilizzato l'algoritmo Particle filtering il quale si occupa di generare i cosiddetti particle, dizionari variabile-valore, mettendo insieme osservazioni con le probabilità calcolate inizialmente e le distribuzioni precedenti. Finita questa operazione vengono calcolate le distribuzioni aggiornate utilizzando i suddetti particle. Dopo ogni avanzamento di stage viene effettuato un resample dei particle, che permette di avere una popolazione di esempi più accurata per la generazione delle future distribuzioni.

4.2 Implementazione algoritmo di localizzazione

L'algoritmo implementato per la localizzazione si occupa di accorpare le informazioni conosciute circa la struttura del grafo, con le distribuzioni di probabilità calcolate per ogni stage dal sistema in questo modo:

In ogni stage il robot si sposta di un nodo sul percorso calcolato attivando i vari sensori che forniscono una osservazione sullo stato del sistema.
Il localizzatore conosce solo la posizione iniziale del robot, per tutti gli stage successivi invece, aggiorna le distribuzioni relative a vari stati del sistema data l'osservazione, dopodiché calcola la posizione più probabile del robot, scegliendo quella, tra i nodi vicini alla precedente posizione calcolata, con la probabilità più alta. Tale posizione diventerà quindi la nuova posizione corrente più probabile e ripete questa sequenza di operazioni per ogni fase dello spostamento del robot.