



PROGETTO 1 BORSA 7PIXEL

Soluzione per la classificazione di accessori e prodotti di Andrea Marisio




28 SETTEMBRE 2016

UNIVERSITÀ DEGLI STUDI DELL'INSUBRIA
Varese


Traccia del progetto

Descrizione della traccia

La traccia scelta tocca principalmente il sito  trovaprezzi.it e qualsiasi negozio online il quale rappresenti gli oggetti in “vetrina” tramite una descrizione testuale.

La traccia è la seguente:

*“Trovare una soluzione originale o provare librerie ed applicazioni che riescono a farci capire in automatico quando abbiamo a che fare con un titolo+descrizione di un prodotto (es. bicicletta) oppure con il titolo+descrizione di un suo accessorio (es. campanello per bicicletta).
Realizzare una demo usando un qualsiasi linguaggio di programmazione.”*

In allegato c'è anche un piccolo dataset di circa 25000 record di cui 12700 accessori e prodotti già classificati. Questi dati sono stati ricavati direttamente dal sito  trovaprezzi.it ed essendo dati pratici sono soggetti ad alcune imperfezioni.

Motivazione della scelta

Durante il mio percorso di tesi presso l'università degli studi dell'Insubria, ho affrontato il problema di creare un prototipo di un'intelligenza artificiale (Context-Aware Bandits) con lo scopo di trovare una soluzione all'onerosità dell'algoritmo. Questa esperienza mi ha aperto le porte del mondo affascinante dell'intelligenza artificiale, una nuova passione.

La traccia alternativa, al momento della scelta, erano solo due: “Distinguere automaticamente i prodotti dai suoi accessori”, questa, e una relativa ad un sistema online di prenotazione. Viste le alternative non ho avuto dubbi, ho scelto questa perché più istruttiva e più vicina ai miei gusti.

Analisi del problema

Prodotto o Accessorio?

Cosa distingue un prodotto da un accessorio? Prima di rispondere a questa domanda prima bisogna chiedersi; Cos'è un accessorio?

Secondo il dizionario Treccani:

*“s. m. Parte o elemento che integra e completa
la funzionalità di qualche cosa.”*

Per poter dire se un record è relativo ad un accessorio bisogna per prima cosa individuare il suo legame con quel “qualche cosa”, che per noi è il prodotto. Purtroppo questa relazione non è sempre esplicita, soprattutto nel caso di accessori standard, come ad esempio degli auricolari i quali, al giorno d’oggi, non c’è bisogno di dire che funzionano con i telefoni (nemmeno del tipo di attacco). Bisogna anche considerare che non si conoscono quali sono i prodotti quindi, anche se la relazione fosse esplicita, sarebbe complesso capire la presenza di tale informazione.

Controllo del Dataset

Il dataset ha avuto un ruolo fondamentale nella scelta della mia strategia di classificazione. Questo era composto da due file “.csv”, uno per i prodotti e uno per gli accessori. Entrambi avevano dei difetti come descrizioni che coincidevano al titolo ripetuto più volte.

Vista la natura del dataset ho scelto di compiere alcuni passi di preprocessing, come la rimozione (sostituzione con “ ” per la precisione) di caratteri non alfanumerici (rimuovendo simboli estetici cose del tipo “**Entro 24h**” oppure eventuali emoticon testuali ecc...) e la rimozione del titolo nelle descrizioni, così da rimuovere la ridondanza informativa.

Il dataset così processato contiene circa 700 oggetti con descrizione vuota.

Visto il grande numero di parole non significative ho ritenuto necessario utilizzare un algoritmo di individuazione delle parole chiave

Ricerca della soluzione

Linguaggio Utilizzato


Ho scelto Java per via della conoscenza pregressa e del codice già posseduto relativo alle reti sociali.

Individuazione delle Features

I dati da cui ricavarne le features sono di tipo testuale, quindi si hanno delle frasi composte da parole e numeri. Ogni parola ha un significato particolare che varia a seconda della posizione nel testo. Per semplificare ho deciso di rappresentare il significato di ogni parola tramite un vettore features, e per semplificare ulteriormente queste features corrispondono geometricamente ad un vettore n-ario (in questo caso binario) che indica il grado di verosimilarità che quella parola compaia in una determinata classe. Mentre per quanto riguarda l'ordine delle parole ho deciso di costruire un grafo da cui estrarre successivamente informazioni sfruttando le tecniche di analisi delle reti sociali (quelle relative per lo più alla struttura), per questioni di tempo, queste non sono state implementate.

Troppe parole!

Una descrizione di un oggetto può essere davvero lunga e complessivamente ricca di parole poco rilevanti. Per risolvere questo problema ho, per prima cosa, cercato un

API che facesse al caso mio, e ho trovato  un'intelligenza artificiale per l'individuazione di keywords, ma vista la mancanza di dati di addestramento (non che questi siano obbligatori), dizionari e la mancanza di chiara documentazione, ho deciso di implementare con soddisfazione un mio sistema.

Il sistema di individuazione di keywords implementato è una cosa semplice semplice, individua le entropie delle parole così da attribuirne un "ranking" di contenuto informativo. Essendo un sistema semplice non raggruppa i verbi coniugati diversamente e non raggruppa le parole che compaiono spesso attaccate (una parte di questa idea è stata implementata, ma sempre per questioni tempistiche non ultimata).

Scelta del tipo di classificatore

Visto il sistema di individuazione delle keyword, ho pensato di utilizzare un classificatore supervisionato di tipo statistico, dove le features di ogni parola venivano addestrate sommando il vettore relativo alla classe di appartenenza e rinormalizzando il tutto.

Difetti

Nel sistema di individuazione delle parole chiave

Il sistema relativo alle keyword presenta dei problemi quando incontra verbi o qualsiasi parola con prefissi/suffissi, poiché le individua come parole diverse e quindi con features distinte

Nel sistema di classificazione

Non può sfruttare le parole che non conosce, volendo si potrebbe attribuire come features di una parola nuova la normalizzazione della somma delle parole dell'oggetto, per poi venire rifinita quando viene incontrata successivamente e sfruttare la similarità dello stato attuale e del vettore che lo rifinisce.

Vari

Ho utilizzato JUnit molto poco, ho fatto pochi test, sicuramente meno del necessario.

Non ho commentato adeguatamente il codice.

Inizialmente volevo utilizzare DTD per la produzione del software, ma vista la mancanza di specifiche (ovvero della presenza di solo dei requisiti) ho preferito fare un semplice e sprovveduto Code&Test.