Aspect-based & opinion mining

Estrazione di caratteristiche e opinioni dalle recensioni online con tecniche di ML

1. Introduzione

Alcuni tipi di documenti, come le recensioni online di prodotti, possono contenere informazioni a grana fine a proposito dei differenti aspetti del prodotto o del servizio cui si riferiscono. Le normali tecniche di Sentiment Analysis permettono di ottenere informazioni di ottima qualità sull'opinione generale di un testo, ma riuscire ad indicare con esattezza quali siano gli specifici aspetti ad essere valutati può essere complesso. Questo compito è reso più semplice dall'**Aspect-Based Sentiment Analysis** (ABSA).
Come indicato da Pavlopoulos [1], non esiste una task decomposition stabilita, né misure di valutazione stabilite per l'ABSA. Perciò, ho deciso di seguire l'approccio proposto da Poria et al. 2016 [2] e di mantenere quest'ultimo durante tutto lo svolgimento del progetto. Esso prevede l'utilizzo, senza precedenti nell'ABSA, di deep convolutional neural network e di tecniche standard di analisi del linguaggio naturale.

2. Descrizione dell'approccio

I passi dell'approccio proposto da [2] sono fondamentalmente tre:

1) L'utilizzo di una rete neurale convoluzionale per etichettare ogni parola dei testi considerati come "aspect word" o "non-aspect word";
2) Lo sviluppo di un insieme di pattern linguistici per lo stesso scopo, da usare in combinazione con la rete neurale;
3) L'accoppiamento dei punti 1 e 2 con un modello word-embedding per la sentiment analysis.

L'utilizzo di una CNN permette di superare le limitazioni tipiche degli altri approcci già usati nell'ambito dell'ABSA, ovvero CRF (Conditional Random Fields) e linguistic patterns; questi approcci sono limitati dalla linearità e dalla scarsa praticità. Lo stesso passo 2 dell'approccio di [2] è limitato dalla necessità di dover specificare manualmente i pattern linguistici.
La struttura della CNN è la seguente:

- un layer di input;
- due convolution layers;
- due max-pool layers;
- un fully connected layer con funzione softmax per l'output.

Come feature, si utilizzano word embeddings addestrati su due diversi corpus di testi. In questo modo, ogni parola viene "codificata" in un vettore di dimensione 300, il quale viene quindi dato in pasto alla rete neurale. Per tale lavoro vengono usati due embedding pre-addestrati: GloVe o Google News Word Embedding.
In aggiunta ai word embeddings, sono di assoluta importanza anche i tag POS, poiché la maggior parte degli aspetti sono rappresentati da nomi; per questo motivo ad ogni parola viene associato un tag POS tra cui nome, verbo, aggettivo, avverbio, preposizione e congiunzione. Questo tag è quindi rappresentato da un vettore 6-dimensionale, che viene concatenato al vettore 300-dimensionale del word embedding. Come POS tagger viene usato lo Stanford Tagger.

In definitiva, il vettore di feature finale è di 306 dimensioni.

Per l'addestramento e la valutazione del modello è stato usato il dataset SemEval 2014. Questo dataset contiene due domini di applicazione, ovvero Laptop e Ristoranti.

3. Implementazione del modello

Per l'implementazione del modello mi sono largamente basato su quella già esistente proposta da [3].
Questa implementazione focalizza l'attenzione sugli aspetti di Deep Learning dell'approccio descritto in [2]; le parti relative alle tecniche standard di NLP, come i pattern linguistici, sono stati tralasciati.

I passi eseguiti sono, nell'ordine:

1) Import delle dipendenze e dei tool;
2) Import di training e test set testuali;
3) Import e indicizzazione dei word embeddings;
4) Tokenizzazione del dataset testuale in un insieme di vettori 2D;
5) Costruzione dell'Embedding layer;
6) Estrazione dei word embeddings dal dataset testuale;
7) Aggiunta delle feature relative ai tag POS;
8) Costruzione e training della CNN;
9) Valutazione del modello in termini di Precision, Recall e F1.

3.1 Import delle dipendenze e dei tool

Come API per la costruzione dei modelli di Deep Learning è stato usato Keras. Altre librerie utili sono state quella per il parsing di file XML e il tagger POS Stanford presente in NLTK.

3.2 Import di training e test set testuali

Come dataset è stato usato quello relativo ai ristoranti di SemEval 2014. L'80% delle recensioni è stato usato come training set, mentre il restante 20% come validation set.

3.3 Import e indicizzazione dei word embeddings

Come word embeddings sono stati usati, separatamente, due dataset pre-addestrati: GloVe e Google News Word Embeddings, quest'ultimo con il supporto di Word2Vec.

3.4 Tokenizzazione del dataset

Questo passo si è reso necessario per l'estrazione dei word embedding dal dataset testuale.

3.5 Costruzione dell'Embedding layer
3.6 Estrazione dei word embeddings

Passi implementati usando le API di Keras.

3.7 Aggiunta dei tag POS

Passo implementato usando il POS Tagger di Stanford; viene usato il One Hot Encoding.

3.8 Training della CNN
Passi implementati usando le API di Keras.

3.9 Valutazione del modello
Il modello è valutato andando a contare il numero di falsi positivi, per poi calcolare le statistiche di Precision, Recall e F1 score.

4. Esecuzione e test del modello
Le prime istanze di esecuzione del modello descritto sopra sono state fatte partire in locale sul mio pc. In effetti, non sono mai riuscito a completare dall'inizio alla fine l'esecuzione di tutti i passi, a causa di molteplici errori, dapprima dovuti a cattiva configurazione, poi da incompatibilità di vario tipo, per finire con problematiche dovute a carenze hardware del mio modesto laptop.
Per ovviare a questa mole di problemi, ho deciso di addestrare ed eseguire il modello in ambiente cloud. Mi sono appoggiato ai servizi di AWS per tale scopo.

4.1 Creazione di una istanza AWS EC2 (Elastic Compute Cloud)
Per usufruire dei servizi offerti da AWS, il primo passo è quello di creare una istanza EC2 su cui far girare il modello. La creazione avviene attraverso un wizard che permette di scegliere piattaforma software e risorse hardware; ho scelto una DLAMI (Deep Learning Amazon Machine Instance) con supporto a Keras 2 e utilizzante un backend TensorFlow su Python 2.7. A livello hardware ho dapprima scelto una semplice istanza t2.micro, per poi passare ad una t2.large leggermente più potente, poiché la computazione risultava parecchio lenta durante alcune fasi (si bloccava durante l'estrazione dei word embeddings a causa dell'esaurimento di memoria).
Una volta creata l'istanza, ho potuto collegarmi tramite SSH ad essa e utilizzare l'interfaccia a linea di comando. Tramite il comando "aws s3 cp" ho recuperato il dataset SemEval 2014 e l'embedding pre-trained GloVe che avevo precedentemente caricato in un bucket del servizio S3 (Simple Storage Service).
Quindi usando ipython, ho potuto addestrare e valutare il modello senza grossi problemi.

4.2 Esecuzione e test del modello
I risultati ottenuti nelle prime esecuzioni del modello sono state del tutto simili, nella valutazione, a quelli ottenuti da [3].
Per il modello che implementa la CNN, il Word Embedding con GloVe e il tagging POS, i risultati sono stati i seguenti:

Precision: 92,7% *Recall*: 56,4% *F1*: 70,1%

5. Tuning degli iperparametri
La valutazione del modello presentava una alta Precision ed una relativamente bassa Recall. Ho voluto effettuare un tuning manuale degli iperparametri per provare ad ottenere una Recall più alta, anche a scapito della Precision. L'idea era quella di ottenere un F1 score ottimale, poiché l'approccio di [2] completo prevede anche l'utilizzo di pattern linguistici, che perfezionano ulteriormente la evaluation del modello, in particolare per quanto riguarda la precision.

5.1 Batch size

Il tuning di questo iperparametro, impostato a 10 inizialmente, non ha comportato un miglioramento né un peggioramento sensibili nella valutazione del modello. Tuttavia, alcuni esperimenti hanno mostrato come raddoppiando questo iperparametro, il tempo necessario per la convergenza è diminuito in modo sensibile (i tempi si riferiscono ad una esecuzione su istanza cloud Amazon t2.large):

Batch size: 10 Tempo per la convergenza dell'addestramento: 178s
Batch size: 20 Tempo per la convergenza dell'addestramento: 152s

5.2 Epoch number

Aumentando la batch size, ho pensato di poter ridurre il numero di epoche, visto che teoricamente la convergenza viene raggiunta più velocemente. Ciò ha effettivamente comportato una ulteriore riduzione del tempo di addestramento, senza alcun cambiamento nei parametri di valutazione del modello.

Epoch number: 50 Tempo per la convergenza: 152s
Epoch number: 40 Tempo per la convergenza: 131s

6. Conclusione

L'approccio proposto da [2] è il primo ad utilizzare tecniche di deep learning, in combinazione con tecniche tradizionali, per la costruzione di un sistema ABSA. L'implementazione da me sperimentata si focalizza sulla parte Deep di questo approccio, realizzando una rete neurale convoluzionale che, combinata con word embeddings pre-addestrati e tagging POS, riesce ad ottenere risultati soddisfacenti; combinare questa implementazione con un modulo di riconoscimento di pattern linguistici completerebbe la realizzazione di un sistema come quello descritto in [2], ottenendo con tutta probabilità le performance descritte nel paper.

7. Riferimenti

[1] "Aspect based sentiment analysis" Pavlopoulos, 2014
[2] "Aspect extraction for opinion mining with a deep convolutional neural network" Poria et al., 2016
[3] https://github.com/jeetp465/Aspect-Based-Sentiment-Analysis