Estrazione degli Embedding da BERT e Clustering con K-Means

Autore: *Tuo Nome*

April 29, 2025

Abstract

In questa relazione descriviamo in dettaglio il processo di estrazione dei vettori (embedding) dal modello BERT `Teto03/Bert_base_fineTuned`, la loro struttura e dimensione, e come vengono utilizzati per il clustering mediante l'algoritmo K-Means.

1 Introduzione

I modelli di linguaggio basati su Transformer, come BERT (Bidirectional Encoder Representations from Transformers), rappresentano ogni input testuale tramite vettori numerici ad alta dimensionalità. Questi vettori, detti *embedding*, contengono informazioni semantiche utili per compiti di classificazione, clustering e molti altri.

2 Tokenizzazione e Input di BERT

Il testo da analizzare viene innanzitutto tokenizzato utilizzando il tokenizer associato al modello. Per ogni sequenza di input:

1. Il tokenizer suddivide il testo in *subword tokens*.
2. Vengono aggiunti i token speciali:
   - `[CLS]` all'inizio della sequenza,
   - `[SEP]` alla fine.
3. L'input viene convertito in ID di token e maschere di attenzione.

3 Estrazione degli Hidden States

Chiamando il modello con l'opzione `output_hidden_states=True`, otteniamo:

$$\texttt{outputs.hidden\_states} = \{H^0, H^1, \ldots, H^L\},$$

dove L è il numero di layer (per `bert-base`, $L = 12$), e ciascun $H^l \in \mathbb{R}^{\text{batch}\times\text{seq\_len}\times d}$, con $d = 768$.

4 Vettore [CLS] come Embedding della Frase

Per rappresentare l'intera sequenza, estraiamo il vettore corrispondente al token `[CLS]` dall'ultimo layer H^L:

$$\mathbf{e} = H^L[:, 0, :] \in \mathbb{R}^d = \mathbb{R}^{768}.$$

Qui la dimensione 768 è specifica del modello `bert-base`. Il vettore $\mathbf{e}$ contiene una rappresentazione compatta dell'intero testo.

5 Costruzione della Matrice di Embedding

Ripetendo il processo per ciascuna delle N risposte caricate da `response.json`, otteniamo un insieme di vettori:

$$\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N\}, \quad \mathbf{e}_i \in \mathbb{R}^{768}.$$

Concatenando verticalmente, si forma la matrice:

$$X = \begin{bmatrix} - & - & - \\ \mathbf{e}_1^T & & \\ - & - & - \\ \mathbf{e}_2^T & & \\ \vdots & & \\ \mathbf{e}_N^T & & \\ - & - & - \end{bmatrix} \in \mathbb{R}^{N\times 768}.$$

6 Clustering con K-Means

L'algoritmo K-Means richiede in input la matrice X. Fissato il numero di cluster K (es. $K = 2$), si procede:

1. Inizializzazione casuale dei centroidi $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}$ in $\mathbb{R}^{768}$.
2. Iterazioni:
   - Assegna ciascun vettore $\mathbf{e}_i$ al cluster di cui il centroide è più vicino (distanza Euclidea):

$$c(i) = \arg\min_{j\in\{1,\ldots,K\}} \|\mathbf{e}_i - \boldsymbol{\mu}_j\|_2.$$

- Aggiorna i centroidi come media dei vettori assegnati:

$$\boldsymbol{\mu}_j = \frac{1}{|C_j|} \sum_{i:c(i)=j} \mathbf{e}_i.$$

3. Convergenza quando le assegnazioni non cambiano più.

7 Riduzione Dimensionale per Visualizzazione

Per rappresentare i dati in 2D:

- **PCA**: proiezione lineare da $\mathbb{R}^{768}$ a $\mathbb{R}^2$ mantenendo la massima varianza.
- **t-SNE**: mappa non lineare che preserva le distanze locali in uno spazio bidimensionale.

Le figure risultanti sono salvate come `clustering_pca.png` e `clustering_tsne.png`.

8 Conclusioni

In questa relazione abbiamo mostrato passo dopo passo come le risposte generate da BERT vengano convertite in vettori densi di dimensione 768 e come questi vengano successivamente soggetti a clustering con K-Means. Le tecniche di riduzione dimensionale permettono una visualizzazione intuitiva dei risultati.