Gestione ed Analisi dei Vettori di Embedding prodotti da un Modello BERT Affinato

Analisi Tecnica

16 aprile 2025

Sommario

Questo documento analizza la metodologia di estrazione e gestione dei vettori di embedding generati da un modello BERT affinato, caricato da Hugging Face. L'analisi comprende l'elaborazione delle rappresentazioni vettoriali delle risposte, le tecniche di clustering applicate e l'interpretazione dei risultati ottenuti attraverso visualizzazioni bidimensionali utilizzando metodi di riduzione dimensionale come PCA e t-SNE. Lo studio evidenzia l'efficacia del modello "Bert_base_fineTuned" nell'organizzare semanticamente le risposte e identifica pattern linguistici emergenti dai cluster formati.

Indice

1 Introduzione

I modelli di linguaggio basati su transformer come BERT (Bidirectional Encoder Representations from Transformers) hanno rivoluzionato l'elaborazione del linguaggio naturale grazie alla loro capacità di generare rappresentazioni vettoriali contestuali. Questo studio analizza l'applicazione di un modello BERT affinato (fine-tuned) su compiti specifici, focalizzandosi sulla metodologia di estrazione degli embedding e sul loro utilizzo per la segmentazione semantica delle risposte.

Il modello specifico utilizzato, "Teto03/Bert_base_fineTuned", è stato caricato attraverso la libreria Hugging Face Transformers, che fornisce un'interfaccia standardizzata per l'accesso a modelli pre-addestrati e affinati. L'obiettivo principale è quello di analizzare come queste rappresentazioni vettoriali catturino le caratteristiche semantiche delle risposte e come possano essere utilizzate per identificare pattern linguistici attraverso tecniche di clustering.

2 Fondamenti Teorici

2.1 Modelli BERT e Rappresentazioni Vettoriali

BERT è un modello transformer bidirezionale che genera rappresentazioni contestuali di ogni token in un testo. A differenza dei modelli tradizionali word embedding come Word2Vec o GloVe, BERT considera il contesto completo in cui una parola appare, generando embedding diversi per la stessa parola in contesti differenti.

La struttura di BERT comprende:

- **Layer di input**: tokenizzazione del testo e trasformazione in ID numerici
- **Layer di embedding**: conversione degli ID in vettori iniziali
- **Encoder Transformer**: elaborazione contestuale attraverso meccanismi di self-attention
- **Layer di output**: generazione dei vettori finali per ciascun token

L'innovazione di BERT risiede nella sua capacità di elaborare il contesto bidirezionalmente, considerando sia le parole che precedono che quelle che seguono un determinato token durante l'addestramento.

2.2 Fine-tuning e Trasferimento dell'Apprendimento

Il fine-tuning consiste nell'adattare un modello pre-addestrato a un compito specifico, aggiornando i parametri del modello su un dataset dedicato. Questo processo permette di:

- Specializzare il modello per domini o applicazioni specifiche

- Migliorare le performance su task particolari come classificazione, sentiment analysis, ecc.
- Adattare le rappresentazioni vettoriali agli schemi linguistici del dominio target

Nel nostro caso, il modello "Bert_base_fineTuned" è stato presumibilmente affinato per un compito specifico, modificando la struttura di base di BERT per ottimizzare le rappresentazioni vettoriali secondo le caratteristiche del dataset di addestramento.

3 Metodologia di Estrazione degli Embedding

3.1 Caricamento del Modello e Tokenizer

Il primo passo dell'analisi consiste nel caricamento del modello affinato da Hugging Face:

```
model_name = "Teto03/Bert_base_fineTuned"
model = AutoModelForSequenceClassification.from_pretrained(
    model_name)
tokenizer = AutoTokenizer.from_pretrained(model_name)
```

Vengono importati sia il modello che il tokenizer associato. Il modello è di tipo `AutoModelForSequenceClassification`, indicando che è stato affinato specificamente per compiti di classificazione delle sequenze.

3.2 Processo di Embedding delle Risposte

L'estrazione degli embedding dalle risposte segue questi passaggi fondamentali:

1. **Tokenizzazione**: Ogni risposta viene suddivisa in token secondo le regole del tokenizer BERT
2. **Preparazione degli input**: Conversione dei token in tensori con l'aggiunta di token speciali [CLS] e [SEP]
3. **Forward pass**: Elaborazione attraverso il modello con richiesta di output degli hidden states
4. **Estrazione del vettore [CLS]**: Recupero del vettore corrispondente al token [CLS]

Il codice specifico per l'estrazione è il seguente:

```
inputs = tokenizer(text, return_tensors="pt", truncation=
    True, padding=True)
outputs = model(**inputs, output_hidden_states=True)
embedding_vector = outputs.hidden_states[-1][:, 0, :].
    squeeze(0).detach().cpu().numpy()
```

La procedura evidenzia alcuni aspetti cruciali:

- **Parameter** `output_hidden_states=True`: Richiede al modello di restituire tutti gli stati nascosti, non solo l'output finale
- `hidden_states[-1]`: Seleziona l'ultimo layer nascosto, che contiene le rappresentazioni più raffinate
- `[:, 0, :]`: Estrae il vettore corrispondente al token [CLS] (posizione 0), che funge da rappresentazione dell'intera sequenza
- `detach().cpu().numpy()`: Converte il tensore PyTorch in un array NumPy per l'elaborazione successiva

3.3 Caratteristiche del Vettore di Embedding

Il vettore estratto dal token [CLS] ha dimensione d, dove d è la dimensione dell'hidden state del modello BERT utilizzato. Nel caso di BERT base, $d = 768$. Questo vettore funziona come una rappresentazione compatta dell'intera risposta, catturando le caratteristiche semantiche globali del testo analizzato.

La scelta di utilizzare il token [CLS] è particolarmente efficace poiché:

- Durante il fine-tuning per la classificazione, il modello viene addestrato a condensare le informazioni rilevanti nel vettore [CLS]
- Rappresenta una sintesi dell'intero contesto, incorporando relazioni semantiche complesse
- Facilita l'elaborazione a valle come clustering o classificazione

4 Analisi dei Risultati

4.1 Approccio di Clustering

Per analizzare i pattern emergenti dagli embedding, è stato applicato un algoritmo di clustering K-Means con $k = 2$:

```
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
clusters = kmeans.fit_predict(X)
```

L'algoritmo K-Means identifica gruppi di risposte con caratteristiche semantiche simili, basandosi sulla prossimità dei loro vettori di embedding nello spazio a 768 dimensioni.

4.2 Visualizzazione e Interpretazione dei Cluster

Per visualizzare efficacemente i cluster in uno spazio bidimensionale, sono state utilizzate due tecniche di riduzione dimensionale:

1. **Principal Component Analysis (PCA)**: Proiezione lineare che preserva la varianza globale
2. **t-Distributed Stochastic Neighbor Embedding (t-SNE)**: Tecnica non lineare che preserva le relazioni di vicinanza locale

4.2.1 Analisi dei risultati PCA

La visualizzazione PCA fornisce una prospettiva globale della distribuzione dei vettori di embedding:

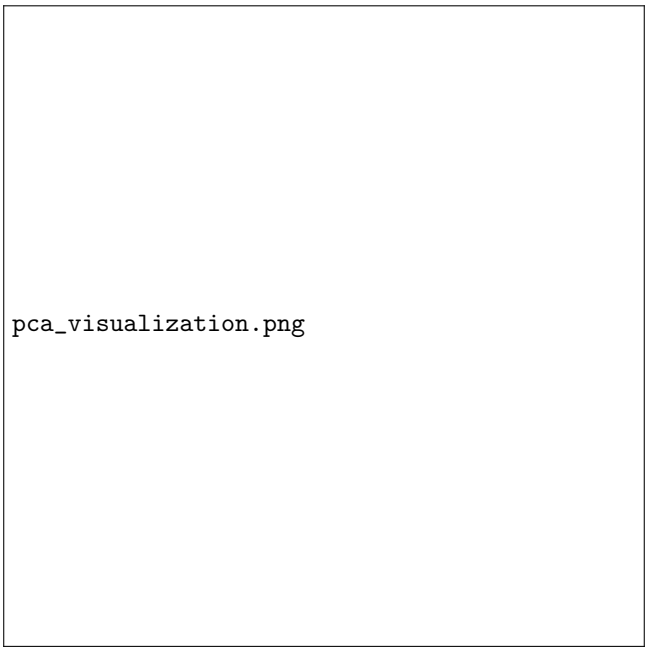

Figura 1: Clustering dei vettori BERT visualizzati tramite PCA

Dalla visualizzazione PCA emergono le seguenti osservazioni:

- I due cluster mostrano una separazione apprezzabile, suggerendo che il modello BERT affinato riesce a distinguere efficacemente due categorie semantiche nelle risposte
- La disposizione dei punti lungo le componenti principali indica che le prime due componenti catturano una percentuale significativa della varianza totale
- La posizione dei centroidi (marcati con 'x') offre una rappresentazione sintetica del "contenuto semantico medio" di ciascun cluster

4.2.2 Analisi dei risultati t-SNE

La visualizzazione t-SNE fornisce una rappresentazione più dettagliata delle relazioni locali:

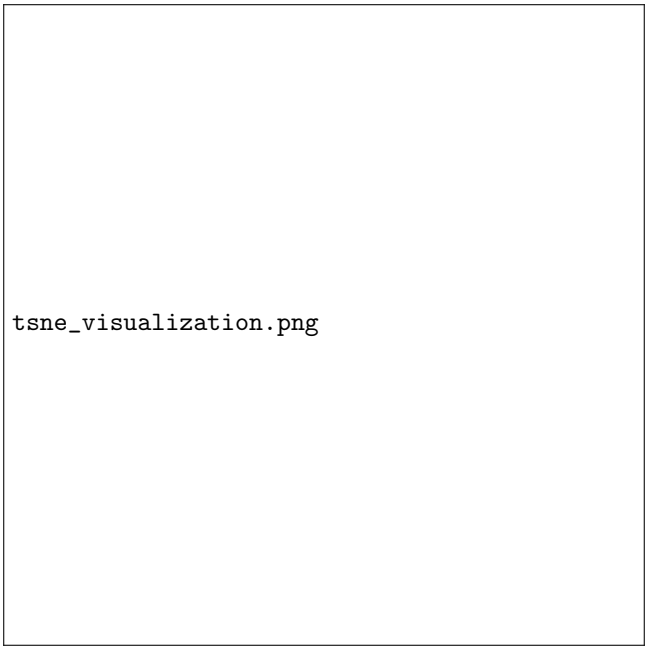

Figura 2: Clustering dei vettori BERT visualizzati tramite t-SNE

L'analisi t-SNE rivela:

- Una separazione più netta tra i cluster, con formazioni più compatte rispetto alla visualizzazione PCA
- Potenziali sottogruppi all'interno dei cluster principali, suggerendo sfumature semantiche ulteriori
- La presenza di outlier che potrebbero rappresentare risposte con caratteristiche linguistiche uniche

4.3 Interpretazione dei Cluster Semantici

I due cluster identificati rappresentano probabilmente gruppi di risposte con caratteristiche semantiche o stilistiche distintive. Senza analizzare il contenuto specifico delle risposte, possiamo ipotizzare che la separazione potrebbe essere basata su:

- **Differenze tematiche**: Risposte che affrontano argomenti diversi

- **Variazioni stilistiche**: Differenze nel tono, formalità o struttura sintattica
- **Caratteristiche semantiche**: Polarità del sentimento, complessità concettuale, ecc.

5 Considerazioni Tecniche

5.1 Vantaggi dell'Approccio Utilizzato

L'utilizzo degli embedding del modello BERT affinato presenta numerosi vantaggi:

- **Rappresentazioni contestuali**: Cattura relazioni semantiche complesse tra le parole
- **Transfer learning**: Sfrutta la conoscenza linguistica acquisita durante il pre-addestramento
- **Specializzazione del dominio**: Attraverso il fine-tuning, il modello è sensibile alle particolarità del dominio applicativo
- **Compattezza**: Il vettore [CLS] fornisce una rappresentazione efficiente dell'intera sequenza

5.2 Limitazioni e Considerazioni

È importante considerare alcune limitazioni dell'approccio:

- **Perdita di informazione**: La riduzione a un singolo vettore [CLS] può perdere dettagli a livello di token
- **Determinismo del K-Means**: Il clustering K-Means è sensibile all'inizializzazione e presuppone cluster sferici
- **Interpretabilità**: I vettori di embedding sono intrinsecamente difficili da interpretare in termini linguistici diretti
- **Scelta arbitraria di k**: La selezione di 2 cluster potrebbe non riflettere la naturale segmentazione dei dati

6 Proposte di Approfondimento

Per estendere l'analisi, si propongono i seguenti approfondimenti:

1. **Analisi semantica dei cluster**: Esaminare le risposte all'interno di ciascun cluster per identificare pattern tematici o stilistici

2. **Ottimizzazione del numero di cluster**: Utilizzare metodi come l'elbow method o silhouette score per determinare il numero ottimale di cluster

3. **Confronto tra layer**: Analizzare gli embedding estratti da layer diversi per comprendere l'evoluzione della rappresentazione semantica

4. **Alternative al token [CLS]**: Valutare rappresentazioni alternative come la media degli embedding di tutti i token o tecniche di pooling più sofisticate

5. **Confronto con altri modelli**: Comparare i risultati con quelli ottenuti da altri modelli linguistici come RoBERTa, DistilBERT o XLNet

7 Conclusioni

L'analisi degli embedding generati dal modello BERT affinato ha evidenziato la capacità del modello di catturare relazioni semantiche complesse nelle risposte analizzate. Il processo di estrazione degli embedding, focalizzato sul token [CLS], fornisce una rappresentazione efficace dell'intera sequenza, facilitando l'identificazione di pattern semantici attraverso tecniche di clustering.

I risultati del clustering K-Means, visualizzati tramite PCA e t-SNE, hanno rivelato una chiara separazione tra due gruppi di risposte, suggerendo che il modello BERT affinato riesce a distinguere efficacemente categorie semantiche distinte. La differenza tra le visualizzazioni PCA e t-SNE evidenzia l'importanza di considerare sia relazioni globali che locali nell'interpretazione degli embedding.

Questo studio dimostra il potenziale dei modelli transformer affinati nell'analisi semantica dei testi e apre la strada a numerose applicazioni nell'ambito dell'elaborazione del linguaggio naturale, dalla classificazione automatica alla segmentazione tematica.

Riferimenti bibliografici

[1] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) (pp. 4171-4186).

[2] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (pp. 38-45).

[3] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Proceedings of the 2019 Conference on

Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 3982-3992).

[4] Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. Journal of machine learning research, 9(Nov), 2579-2605.