Modulo 3: Classificazione del Testo e Analisi del Sentiment 🏷️😊

Corso di Natural Language Processing

📋 Contenuti del modulo

- 🏷️ **Classificazione del testo**: concetti fondamentali
- 🧮 **Approcci tradizionali**: Naive Bayes, SVM, Logistic Regression
- 🧠 **Approcci neurali**: RNN, CNN, Transformer
- 😊 **Analisi del sentiment**: teoria e sfide specifiche
- 💼 **Applicazioni pratiche** in contesti aziendali
- 🛠️ **Implementazione e considerazioni etiche**

🏷️ Classificazione del testo: concetti fondamentali

> "La classificazione del testo è il processo di assegnazione di categorie predefinite a documenti testuali."

- 📝 **Input**: documento testuale (email, recensione, articolo, tweet...)
- 🏷️ **Output**: una o più categorie/etichette
- 🎯 **Obiettivo**: generalizzare dai dati di addestramento a nuovi documenti
- 🔄 **Processo**: supervisionato (richiede esempi etichettati)

🏷️ Tipi di problemi di classificazione

- 0️⃣1️⃣ **Classificazione binaria**: due classi (spam/non-spam)
- 🔢 **Classificazione multi-classe**: più classi mutuamente esclusive (categorie di notizie)
- 🏷️ **Classificazione multi-label**: più etichette contemporaneamente (tag di un articolo)
- 📊 **Classificazione gerarchica**: categorie organizzate in struttura ad albero

🧮 Pipeline di classificazione del testo

1. 📥 **Raccolta dati** etichettati
2. 🧹 **Preprocessing** del testo
3. 🔍 **Feature extraction**
4. 🧠 **Addestramento** del modello
5. 📊 **Valutazione** delle performance
6. 🚀 **Deployment** in produzione

🧹 Preprocessing e Feature Extraction

✂️ Preprocessing

- 🔤 **Tokenizzazione** → dividere il testo in parole o token
- 📝 **Normalizzazione** → minuscolo, rimozione punteggiatura
- 🚫 **Rimozione stopwords** → eliminare parole comuni (“il”, “la”, “e”...)
- 🌱 **Stemming / Lemmatizzazione** → ridurre alla forma base (es. “correndo” → “correre”)

🔍 Feature Extraction

- 🧮 **Bag-of-Words (BoW)** → conta quante volte appaiono le parole
- 📊 **TF-IDF** → pesa parole frequenti nel testo ma rare nei documenti
- 🔤 **N-grams** → sequenze di N parole consecutive
- 🧠 **Word Embeddings** → vettori densi (Word2Vec, GloVe, FastText)
- 🔄 **Embeddings contestuali** → modelli avanzati (BERT, RoBERTa) che capiscono il contesto

ℹ️ Note importanti

- Il preprocessing “pulisce” il testo per prepararlo all’analisi
- La feature extraction trasforma il testo in numeri per i modelli
- Si è passati da metodi semplici e sparsi (BoW, TF-IDF) a metodi densi e complessi (embeddings)
- ⚠️ Esempio:
  BoW → “il cane morde l’uomo” ≈ “l’uomo morde il cane”
  BERT → distingue l’ordine e il significato

💬 Domanda possibile

👉 In quali casi conviene usare **BoW** invece di embeddings avanzati?

🧮 Approcci tradizionali: Naive Bayes

- 📊 Basato sul teorema di Bayes con assunzione di indipendenza
- 🧮 $P(y|x) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$
- 🚀 Veloce, efficiente, poco costoso computazionalmente
- 👍 Funziona sorprendentemente bene per classificazione testuale

Varianti:

- 🔢 **Multinomial NB**: conta le occorrenze (per BoW)
- 0️⃣1️⃣ **Bernoulli NB**: presenza/assenza di feature (per testi brevi)
- 📈 **Gaussian NB**: per feature continue

🧮 Approcci tradizionali: Support Vector Machine (SVM)

- 📏 Trova l'iperpiano ottimale che separa le classi
- 🔍 Massimizza il margine tra le classi
- 💪 Molto efficace per testi, specialmente con feature TF-IDF
- ⚠️ Può essere lento su dataset molto grandi

🧮 Approcci tradizionali: Logistic Regression

- 📊 Modello lineare per classificazione probabilistica
- 🧮 $P(y=1|x) = \frac{1}{1+e^{-w^T x}}$
- 🎯 Ottimizza i pesi per massimizzare la verosimiglianza
- 👍 Semplice, interpretabile, efficace
- 🔢 Fornisce probabilità (non solo etichette)
- 🛠️ Facilmente estendibile a classificazione multi-classe

🧮 Approcci tradizionali: Random Forest e Gradient Boosting

- 🌲 **Random Forest**: ensemble di alberi decisionali
  - 🔀 Addestra molti alberi su sottoinsiemi casuali di dati e feature
  - 🗳️ Combina le previsioni tramite voto di maggioranza
  - 🛡️ Robusto all'overfitting, gestisce bene feature irrilevanti
- 📈 **Gradient Boosting**: costruisce modelli sequenzialmente
  - 🔄 Ogni nuovo modello corregge gli errori dei precedenti
  - 💪 Spesso ottiene performance state-of-the-art (XGBoost, LightGBM)
  - ⚙️ Richiede tuning attento degli iperparametri

🧠 Approcci neurali: Reti Neurali Feed-Forward

- 🧩 Input: rappresentazioni vettoriali del testo (BoW, TF-IDF, embeddings)
- 🔄 Hidden layers con attivazioni non lineari
- 🎯 Output layer con softmax per probabilità di classe
- 👍 Può catturare pattern complessi
- ⚠️ Richiede più dati rispetto ai modelli classici
- 🔍 Non cattura naturalmente la sequenzialità del testo

🧠 Approcci neurali: Reti Neurali Ricorrenti (RNN)

- 🔄 Processano sequenze elemento per elemento, mantenendo uno stato nascosto
- 🧠 Varianti avanzate: LSTM (Long Short-Term Memory) e GRU (Gated Recurrent Unit)
- 👍 Catturano dipendenze sequenziali e contestuali
- 📏 Gestiscono input di lunghezza variabile
- ⚠️ Addestramento più complesso, problemi di gradienti svanescenti
- 🐢 Processamento sequenziale lento (non parallelizzabile)

🧠 Approcci neurali: Reti Neurali Convoluzionali (CNN)

- 🔍 Applicano filtri convoluzionali per catturare pattern locali
- 🧩 Architettura tipica: embedding → convoluzione → max-pooling → fully connected
- 👍 Efficaci per catturare n-grammi e pattern locali
- 🚀 Più veloci da addestrare rispetto alle RNN
- 🔄 Non catturano dipendenze a lungo termine come le RNN
- 💪 Sorprendentemente efficaci per la classificazione testuale

🧠 Approcci neurali: Transformer

- ⚡ Basati sul meccanismo di self-attention
- 🔄 Processano l'intera sequenza in parallelo
- 🌐 Pre-addestrati su enormi corpora (BERT, RoBERTa, XLNet...)
- 🎯 Fine-tuning per task specifici di classificazione
- 🏆 State-of-the-art per la maggior parte dei task NLP
- ⚠️ Computazionalmente costosi, richiedono GPU
- 📏 Limitazioni sulla lunghezza dell'input

Analisi del sentiment

😊 Analisi del sentiment: concetti fondamentali

> "L'analisi del sentiment è il processo di determinazione dell'attitudine, opinione o emozione espressa in un testo."

- 🎯 **Obiettivo**: identificare e quantificare il sentiment espresso
- 📊 **Granularità**: documento, frase, aspetto, entità
- 🔍 **Approcci**: basati su lessico, machine learning, ibridi
- 🧠 **Output**: categorico (pos/neg/neutro) o continuo (score)

😊 Livelli di analisi del sentiment

- 📝 **Livello documento**: sentiment globale dell'intero documento
  - "Questo prodotto è fantastico. Altamente consigliato!"
- 📋 **Livello frase**: sentiment di singole frasi
  - "L'interfaccia è intuitiva, ma la batteria si scarica velocemente."
- 🎯 **Livello aspetto**: sentiment verso specifici aspetti/caratteristiche
  - "La fotocamera è eccellente [+], ma il prezzo è troppo alto [-]."
- 🏢 **Livello entità**: sentiment verso specifiche entità
  - "Apple ha rilasciato un ottimo prodotto, ma Samsung resta leader."

😊 Approcci all'analisi del sentiment

- 📚 **Approcci basati su lessico**:
  - Utilizzano dizionari di parole con polarità predefinite
  - Es: VADER, SentiWordNet, AFINN
  - 👍 Non richiedono addestramento
  - 👎 Limitati da espressioni complesse, sarcasmo, contesto
- 🧠 **Approcci basati su machine learning**:
  - Supervisionati: Naive Bayes, SVM, deep learning
  - 👍 Catturano pattern complessi
  - 👎 Richiedono dati etichettati

😊 Sfide nell'analisi del sentiment

- 🎭 **Sarcasmo e ironia**: "Fantastico, un altro aggiornamento che rallenta tutto!"
- 🔄 **Negazioni**: "Il prodotto non è male" (positivo, non negativo)
- 📏 **Intensificatori**: "molto buono" vs "buono"
- 🌐 **Espressioni idiomatiche**: "costare un occhio della testa"
- 🧩 **Ambiguità**: "Il film era incredibile" (positivo o negativo?)
- 🌍 **Differenze culturali e linguistiche**: variazioni nell'espressione di emozioni
- 📱 **Emoji e emoticon**: 😊 vs 🙄 (richiedono interpretazione specifica)

📊 Valutazione dei classificatori testuali

- ✅ **Accuracy**: proporzione di previsioni corrette
  - $\text{Accuracy} = \frac{\text{Corrette}}{\text{Totale}}$
- 📊 **Precision**: proporzione di positivi identificati correttamente
  - $\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$
- 📈 **Recall**: proporzione di positivi reali identificati
  - $\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$
- 🔍 **F1-Score**: media armonica di precision e recall
  - $\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

📊 Valutazione: Confusion Matrix

- 📊 Tabella che mostra le previsioni vs. realtà
- 🔍 Rivela pattern di errori specifici
- 🧩 Particolarmente utile per classi sbilanciate
- 📈 Base per calcolare precision, recall, F1
- 🔄 Essenziale per comprendere il comportamento del modello oltre le metriche aggregate

📊 Valutazione: ROC Curve e AUC

- 📈 **ROC**: Receiver Operating Characteristic
- 🔍 Mostra il trade-off tra true positive rate e false positive rate
- 📊 **AUC**: Area Under the Curve
  - 1.0 = classificatore perfetto
  - 0.5 = classificatore casuale
- 👍 Robusta rispetto a classi sbilanciate
- 🔄 Utile per confrontare modelli e scegliere soglie di decisione

💼 Caso di studio: Analisi delle recensioni dei clienti

Azienda: Catena di hotel internazionale

Sfida: Analizzare migliaia di recensioni per identificare punti di forza e debolezza

Soluzione:

1. 🧹 Preprocessing delle recensioni
2. 😊 Analisi del sentiment a livello di aspetto (pulizia, personale, posizione, prezzo...)
3. 📊 Dashboard con trend temporali e confronto tra strutture
4. 🔍 Alert automatici per problemi ricorrenti

Risultato: Miglioramento del 15% nella soddisfazione dei clienti in 6 mesi

💼 Caso di studio: Monitoraggio della reputazione del brand

Azienda: Produttore di elettronica di consumo

Sfida: Monitorare la percezione del brand sui social media e forum

Soluzione:

1. 🔍 Raccolta continua di menzioni del brand
2. 🏷️ Classificazione per tema (qualità, prezzo, supporto, innovazione...)
3. 😊 Analisi del sentiment per tema
4. 📊 Dashboard in tempo reale con alert

Risultato: Identificazione precoce di una potenziale crisi PR, con risposta rapida che ha limitato l'impatto negativo

💼 Caso di studio: Classificazione automatica di ticket di supporto

Azienda: Software as a Service (SaaS)

Sfida: Smistare automaticamente migliaia di ticket di supporto

Soluzione:

1. 🧠 Classificatore multi-classe basato su BERT
2. 🎯 Categorizzazione in 20+ categorie (bug, domande di fatturazione, richieste di feature...)
3. 🔄 Sistema di feedback per miglioramento continuo
4. 👥 Integrazione con sistema di assegnazione agli specialisti

🛠️ Implementazione pratica: considerazioni tecniche

- 🔄 **Pipeline end-to-end**: dalla raccolta dati al deployment
- 🚀 **Scalabilità**: gestione di volumi crescenti di dati
- ⏱️ **Latenza**: tempo di risposta accettabile per l'applicazione
- 🛡️ **Robustezza**: gestione di input anomali e casi edge
- 📊 **Monitoraggio**: tracking delle performance nel tempo
- 🔄 **Feedback loop**: meccanismi per miglioramento continuo
- 💰 **Costi**: bilanciamento tra performance e risorse computazionali

🛠️ Considerazioni etiche e bias

- 🧠 **Bias nei dati di addestramento**: riflettono pregiudizi umani
- 🔍 **Trasparenza algoritmica**: comprensione delle decisioni
- 🛡️ **Privacy**: gestione di dati sensibili o personali
- 🌐 **Fairness**: equità tra gruppi demografici
- 🔄 **Accountability**: responsabilità per le decisioni automatizzate
- 📊 **Audit regolari**: verifica continua di bias e performance
- 👥 **Human-in-the-loop**: supervisione umana per decisioni critiche

🛠️ Tecniche di interpretabilità

- 🔍 **Feature importance**: quali parole influenzano maggiormente la decisione
- 📊 **LIME (Local Interpretable Model-agnostic Explanations)**: spiega singole predizioni
- 🧮 **SHAP (SHapley Additive exPlanations)**: contributo di ogni feature
- 👁️ **Attention visualization**: visualizzazione dei pesi di attenzione
- 🔄 **Counterfactual explanations**: "cosa cambierebbe la predizione?"
- 📝 **Spiegazioni in linguaggio naturale**: traduzione delle decisioni in testo comprensibile

🛠️ Strumenti e librerie per classificazione testuale

- 🧰 **Scikit-learn**: implementazioni di algoritmi classici
- 🧠 **TensorFlow/Keras e PyTorch**: framework per deep learning
- 🤗 **Hugging Face Transformers**: accesso a modelli pre-addestrati
- 😊 **NLTK e TextBlob**: strumenti per analisi del sentiment basata su lessico
- 🔍 **spaCy**: pipeline NLP end-to-end
- 📊 **LIME e SHAP**: strumenti per interpretabilità dei modelli
- 🧪 **MLflow**: tracking di esperimenti e gestione di modelli

💡 Concetti chiave da ricordare

- 🏷️ La classificazione testuale è un task supervisionato fondamentale nell'NLP
- 🧮 Algoritmi classici (NB, SVM) sono ancora rilevanti per molti casi d'uso
- 🧠 Deep learning offre performance superiori ma richiede più dati e risorse
- 😊 L'analisi del sentiment può essere applicata a diversi livelli di granularità
- 📊 La valutazione deve considerare metriche appropriate al contesto
- 💼 Le applicazioni aziendali spaziano dal customer service al brand monitoring
- 🛠️ Implementazione e considerazioni etiche sono cruciali per il successo

📚 Risorse per approfondire

- 📖 **Paper**: "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" (Devlin et al., 2019)
- 🧪 **Dataset**: IMDB Reviews, Amazon Reviews, Twitter Sentiment
- 🎓 **Tutorial**: "Text Classification with BERT" (TensorFlow)
- 🧰 **Librerie**: VADER per sentiment analysis, Hugging Face per classificazione
- 📊 **Competizioni**: Kaggle text classification challenges
- 📚 **Libro**: "Natural Language Processing with Transformers" (Lewis et al., 2021)

🙋 Domande?

Prossimo modulo: Modelli Linguistici e Sequence-to-Sequence

Grazie per l'attenzione! 👋