UNIVERSITÀ
DI PARMA

DIPARTIMENTO DI INGEGNERIA E ARCHITETTURA

Corso di Laurea Ingegneria Informatica, Elettronica e delle Telecomunicazioni

Classificazione automatica delle emozioni con BERT applicata a Stack Overflow

Emotion detection with BERT applied to Stack Overflow

Relatore:
Prof. Michele Tomaiuolo

Tesi di Laurea di:
Matteo Gianvenuti
Matricola: 321490

Anno Accademico 2022/2023

Ringraziamenti

Ringrazio il Prof. Michele Tomaiuolo, relatore della tesi, per l'aiuto fornito nella realizzazione di questa ricerca e tutti coloro che mi hanno supportato in questo percorso.

Indice

Introduzione 5
1. Background 6
1.1 Stack Overflow 6
1.2 Linguaggi di programmazione 7
1.3 Intelligenza artificiale e machine learning 7
1.3.1 Apprendimento non supervisionato 8
1.3.2 Apprendimento supervisionato 9
1.4 Distant supervision 9
1.5 Natural Language Processing 10
1.6 Bidirectional Encoder Representations Transformers 10
1.6.1 Transformers 11
1.7 Overfitting 12
1.8 Sentiment analysis e emotion detection 13
1.9 Classificazione delle emozioni 14
2. Architettura 15
2.1 Knowledge Discovery in Databases (KDD) 15
2.2 Come e dove sono stati reperiti i dati 17
2.3 Preprocessing 17
2.4 Trasformazione dei dati e creazione dei dataset 17
2.5 Applicazione del classificatore 18
3. Implementazione 19
3.1 Approfondimento sulla selezione dei dati 19
3.2 Organizzazione dei dati e preprocessing 20
3.3 Approfondimento sulla trasformazione dei dati e distant supervision 21
3.3.1 Trasformazione dei dati 21
3.3.2 Distant supervision (algoritmo utilizzato) 22
3.4 Approfondimento sulla creazione dei dataset 24
3.4.1 Filtraggio dei linguaggi di programmazione 24
3.4.2 Divisione del dataset 25
3.4.3 Bilanciamento dei dataset 26

3.5 Approfondimento sul classificatore 32
3.5.1 Rappresentazione del dataset 32
3.5.2 Preparazione dei dati 33
3.5.3 Addestramento e predizioni del modello 33
3.5.4 Prestazioni del modello 36
3.5.5 Applicazione del modello a SO (Big data analytics) 38
4. Risultati 39
4.1 Tecnologie più popolari secondo il modello 39
4.2 Tecnologie più amate e temute secondo SO 41
4.3 Emozioni predette dal modello verso le tecnologie più popolari 43
5. Conclusione 52
Bibliografia 53

Indice delle figure

Figure 1: Architettura del trasformatore. 12
Figure 2: L'eccessivo addestramento porta ad un errore maggiore sul test set. 13
Figure 3: Classificazione delle emozioni secondo Parrott. 14
Figure 4: Il processo di KDD. 16
Figure 5: Tecnologie più popolari secondo Stack Overflow Developer Survey 2022. 24
Figure 6: Definizione di dizionari, modello e tokenizer. 32
Figure 7: Classe SimpleDataset, rappresenta un dataset generico. 33
Figure 8: Preparazione del training set. 33
Figure 9: Argomenti di addestramento. 35
Figure 10: Definizione del Trainer e addestramento del modello. 36
Figure 11: Correlazione tra le predizioni del modello ed il sondaggio di SO. 42
Figure 12: Emozioni predette dal modello verso Python. 44
Figure 13: Emozioni predette dal modello verso JavaScript. 44
Figure 14: Emozioni predette dal modello verso Java. 45
Figure 15: Emozioni predette dal modello verso C#. 45
Figure 16: Emozioni predette dal modello verso R. 46
Figure 17: Emozioni predette dal modello verso C++. 46
Figure 18: Emozioni predette dal modello verso HTML. 47
Figure 19: Emozioni predette dal modello verso PHP. 47
Figure 20: Emozioni predette dal modello verso SQL. 48
Figure 21: Emozioni predette dal modello verso TypeScript. 48
Figure 22: Emozioni predette dal modello verso CSS. 49
Figure 23: Emozioni predette dal modello verso C. 49
Figure 24: Emozioni predette dal modello verso Swift. 50
Figure 25: Emozioni predette dal modello verso Kotlin. 50
Figure 26: Emozioni predette dal modello verso Dart. 51
Figure 27: Emozioni predette dal modello verso PowerShell. 51

Introduzione

La sentiment analysis o opinion mining è l'uso dell'elaborazione del linguaggio naturale (NLP) e dell'analisi del testo che si occupa di costruire sistemi informatici per estrarre opinioni dal testo, quindi quantificare e studiare gli stati emotivi e le informazioni soggettive di un testo. Negli ultimi anni l'analisi del sentimento è diventata sempre più importante, specialmente nel campo dell'intelligenza artificiale e delle scienze sociali grazie al ruolo che ricopre nella società.

Al giorno d'oggi le opinioni di altre persone, in particolare l'opinione pubblica, sono sempre più influenti nelle scelte di un individuo. Questo andamento riguarda anche i linguaggi di programmazione, infatti, sono sempre più presenti "social network tecnici", forum, blog o gruppi di discussione dove si discute sui linguaggi di programmazione. Spesso si arriva a discussioni agitate ed emotive dove si dimentica che i linguaggi di programmazione hanno pregi e difetti dipendenti dal caso d'uso.

L'obiettivo di questa ricerca è quello di realizzare un modello di intelligenza artificiale quindi di machine learning (ML) per il NLP tramite l'algoritmo BERT basato sui transformer per l'analisi del testo, pre-addestrato su 3300 milioni di parole, che sia capace di individuare quali sentimenti vengono espressi, sulle piattaforme di discussione tecnica, nei confronti dei più popolari linguaggi di programmazione e delle tecnologie ad essi associate nel 2022 secondo il sondaggio Stack Overflow Developer Survey 2022, dove sono stati intervistati più di 70.000 sviluppatori. L'obiettivo è anche quello di osservare come variano nel corso dell'anno 2022 le emozioni espresse verso i linguaggi di programmazione. In particolare, questa ricerca è stata fatta sui post di Stack Overflow nel 2022. Essendo uno dei più grandi siti di domande e risposte sui linguaggi di programmazione è un'ottima risorsa per questo studio. Il sito mette periodicamente a disposizione sotto licenza comune creativa i contenuti pubblici creati dagli utenti.
Per individuare le diverse sfumature delle emozioni espresse la sola polarità della frase non è sufficiente, la classificazione quindi è basata sulla suddivisione delle emozioni di Parrott con l'aggiunta dell'emozione neutrale. Avendo noto che i classificatori addestrati su social network "classici" hanno basse prestazioni su social network "tecnici" come Stack Overflow, è stato realizzato un dataset con i dati rilasciati da Stack Overflow per il 2022, in particolare 3.154.807 domande e risposte (post). In questa ricerca sono stati etichettati i post per l'addestramento del modello con un processo automatizzato di *distant supervision*. Per la classificazione viene scelto il modello migliore dopo tre epoche di addestramento sui dati di allenamento.

1. Background

1.1 Stack Overflow

Stack Overflow (SO) è un forum completamente dedicato ai linguaggi di programmazione quindi agli sviluppatori. In particolare, si tratta di un sito di domande e risposte creato nel 2008 da Joel Spolsky e Jeff Atwood con l'idea di rendere le conoscenze di programmazione più accessibili a tutti. SO fa parte della rete di domande e risposte Stack Exchange (SE). [1]
Per visualizzare domande e risposte (post) non è necessario avere un account registrato mentre è obbligatorio per pubblicare commentare e votare i post, inoltre, il sito dà un punteggio agli utenti in base alla loro attività, questo dà accesso ad ulteriori funzionalità. I post su SO riguardano un'ampia gamma di argomenti relativi alla programmazione concentrandosi però su uno specifico problema. Oltre alla programmazione i post possono riguardare le tecnologie ad essa correlate. Per mantenere un'elevata qualità dei contenuti ai post è associato un punteggio, se questi ricevono un basso punteggio vengono rimossi, anche per evitare contenuti fuorvianti, al contrario se ricevono un elevato punteggio vengono mostrati per primi in modo da dare risalto ai contenuti ritenuti più utili dalla comunità. Alle domande sono associati dei tag, questo può essere molto utile per identificare i linguaggi di programmazione, le tecnologie o i problemi di proprio interesse.
Il sito è molto popolare, da una veloce ricerca browser di un problema si trovano subito tutte le domande e relative risposte ad esso correlate.
Infatti, SO vanta una comunità molto attiva, oltre 100 milioni di visitatori mensili, a marzo 2022 Stack Overflow ha oltre 20 milioni di utenti registrati e ha ricevuto oltre 24 milioni di domande e 35 milioni di risposte. Un utente che pubblica una domanda non deve aspettare a lungo per l'intervento di altri utenti. Infatti, già nel 2011 il 92 % delle domande aveva ricevuto risposta, la quale in media viene fornita dopo un tempo di 11 minuti.
SE rilascia periodicamente una copia del contenuto generato dagli utenti sotto licenza (Creative Common License CC BY-SA 2.5, 3.0 e 4.0) e questa è un'ottima fonte per effettuare studi sulle emozioni verso i linguaggi di programmazione. [2]

1.2 Linguaggi di programmazione

Un linguaggio di programmazione è un sistema di notazione per una sequenza o un insieme di istruzioni che un computer deve eseguire.
Un linguaggio di programmazione è solitamente suddiviso nelle due componenti: sintassi (forma) e semantica (significato). La sintassi è quell'insieme di regole che definiscono le combinazioni di simboli che sono considerate affermazioni o espressioni correttamente strutturate per il linguaggio; mentre la semantica assegna un significato computazionale a quell'insieme di simboli validi definiti nella sintassi del linguaggio di programmazione.
Spesso sintassi e semantica formano un insieme di regole e conoscenze vasto, per questo imparare un linguaggio di programmazione nel suo complesso è un processo lungo e dispendioso che può richiedere molto tempo e impegno. È quindi molto importante in un qualsiasi progetto di sviluppo software la scelta del linguaggio di programmazione. La scelta non dipende solo dal dominio di applicazione, i problemi da risolvere e i vantaggi/ svantaggi di ogni linguaggio ma dipende anche dalle opinioni degli altri sviluppatori. Di conseguenza la sentiment analysis verso i linguaggi di programmazione risulta essere molto importante poiché dalle emozioni degli sviluppatori dipende anche la qualità del software che sviluppano.
Per aiutare gli sviluppatori a capire quali linguaggi sono più usati dalla comunità stessa degli sviluppatori vengono realizzate molte classifiche in base a varie metriche. [3]

In questa ricerca come classifica o sondaggio di riferimento, per i linguaggi di programmazione più popolari nel 2022, è stata usata la classifica annuale realizzata da SO intervistando oltre 70.000 sviluppatori.

1.3 Intelligenza artificiale e machine learning

L'intelligenza artificiale (AI) è una disciplina che studia e implementa metodi per realizzare sistemi informatici "intelligenti" che cercano di simulare, eguagliare o superare le capacità di pensiero e ragionamento umano. AI ha svariate applicazioni come motori di ricerca avanzati, auto a guida autonoma, in campo medico, in campo finanziario nella comprensione del linguaggio naturale e tante altre. In questo lavoro di ricerca viene applicata all'elaborazione e comprensione del linguaggio naturale o umano (NLP) per comprendere quali emozioni sono presenti in un testo. Man mano che le "macchine" diventano sempre più capaci sostituisco il lavoro dell'uomo in molti ambiti poiché sono in grado di effettuare elaborazioni che richiederebbero molto tempo e risorse all'uomo.

Il problema generale della simulazione o creazione dell'intelligenza è stato scomposto in sotto-problemi quali rappresentazione della conoscenza, apprendimento automatico, NLP e intelligenza generale. [4]

Machine learning (ML) o apprendimento automatico è un'ampia sottobranca dell'intelligenza artificiale che si occupa della comprensione e costruzione di metodi che consentono alle macchine di "imparare". Si tratta quindi di metodi che sfruttano grandi quantità di dati per migliorare le prestazioni di computer su una serie di attività. il ML raccoglie metodi scientifici come data mining, statistica computazionale, riconoscimento di pattern, reti neurali artificiali, eccetera.
Gli algoritmi di ML costruiscono un modello basato su un insieme di dati di addestramento per fare previsioni senza essere esplicitamente programmati per farlo. Alcuni algoritmi come, ad esempio, BERT oltre ad avere un set di dati per l'addestramento ne hanno uno per la convalida che viene usato sempre in fase di addestramento per fare una sorta di test anticipato in cui si verifica che il modello sta apprendendo "bene", cioè in modo generalizzato senza imparare a memoria il set di addestramento, ovvero per evitare l'overfitting. Il modello costruito nella fase di addestramento viene poi utilizzato per fare predizioni su dati nuovi, mai visti dal modello. [5]

1.3.1 Apprendimento non supervisionato

L'apprendimento non supervisionato o unsupervised learning (UL) è una tecnica di apprendimento automatico (ML) che impara da dataset non etichettati per "istruire" un sistema informatico. L'obiettivo è che attraverso il mimetismo, che è un importante modo di apprendere anche per le persone, il sistema informatico sia costretto a costruire una rappresentazione conscia dell'ambiente che sta apprendendo. In particolare, consiste nel fornire al sistema una serie di input (esperienza del sistema) che dovrà riclassificare ed organizzare sulla base di caratteristiche comuni per cercare di fare ragionamenti e di conseguenza previsioni sugli input successivi. Si utilizza quando le classi che si vuole identificare non sono note a priori e devono essere apprese automaticamente.
Le tecniche di ML non supervisionato lavorano confrontando i dati e ricercando similarità o differenze. Utilizzano principalmente tecniche di statistica, di conseguenza sono molto efficienti con elementi di tipo numerico o comunque traducibili in numeri. Se i dati non hanno un ordinamento intrinseco può essere difficile avere buone prestazioni.
Gli algoritmi di UL possono risolvere problemi più complessi rispetto agli algoritmi di ML supervisionati, però talvolta possono avere comportamenti più imprevedibili. [6]

Un esempio dell'utilizzo di UL è il pre-addestramento di BERT, eseguito su grandi quantità di dati quali tutta Wikipedia inglese e il BookCorpus. Questo tipo di addestramento fornisce a BERT una conoscenza e comprensione generale della lingua.

1.3.2 Apprendimento supervisionato

L'apprendimento supervisionato o supervised learning (SL) è sempre una tecnica di ML che mira a istruire un sistema informatico per consentirgli di fare previsioni sui valori in uscita dati gli input. In questo caso però gli esempi di addestramento sono etichettati, l'addestramento viene quindi eseguito su un insieme formato da coppie di input e output. Il modello deve imparare a prevedere, tramite ipotesi induttiva, l'output dato l'input.
Un algoritmo di SL deve, in fase di addestramento, analizzare i dati etichettati in input e ricavare una funzione o un modello che può essere utilizzato per mappare nuovi esempi. Nella fase di addestramento bisogna, come nel UL, generalizzare bene per evitare che il modello impari a memoria il dataset di addestramento, cioè, evitare l'overfitting. [7]

Un esempio di SL è l'addestramento di BERT, successivo al pre-addestramento, dove viene specializzato il modello per un task specifico grazie ad un dataset etichettato.

1.4 Distant supervision

La maggior parte delle tecniche di machine learning richiede una serie di dati di addestramento etichettati.
Un approccio tradizionale per la raccolta dei dati di addestramento prevede l'etichettatura manuale di una serie di documenti, questo tipo di approccio può essere molto oneroso nel caso in cui il dataset di addestramento sia molto grande, come ad esempio lo è un dataset costituito da tutti i post su SO nel 2022.
Un approccio alternativo all'etichettatura manuale dei dati è la *distant supervision* o supervisione a distanza, dove tramite un algoritmo il processo viene automatizzato. In particolare, si definiscono una serie di "regole" che l'algoritmo utilizza per stabilire a quale classe appartiene un elemento. Successivamente l'algoritmo valuta ogni elemento del dataset e stabilisce per ognuno di questi un'etichetta. In questo modo si può ottenere facilmente una grande quantità di dati etichettati per l'addestramento di un modello di ML. Tuttavia, questi dati possono essere rumorosi ma essendo il dataset molto grande non ha importanza poiché la maggior parte degli esempi sarà etichettata correttamente. Per citare gli studenti di Stanford che hanno dato questa definizione la distant supervision è più un'arte che una scienza. [8]

1.5 Natural Language Processing

Natural Language Processing (NLP) è una sotto branca interdisciplinare di linguistica, informatica e intelligenza artificiale (AI) che si occupa dell'interazione tra computer e linguaggio naturale o linguaggio umano. In particolare, utilizza algoritmi per analizzare ed estrarre il contenuto di documenti in linguaggio naturale con l'obiettivo di far "comprendere" il contenuto dei documenti ad un computer, comprese le sfumature contestuali della lingua, nonché classificare e categorizzare i documenti stessi.
Questo processo è difficile e complesso a causa delle caratteristiche intrinseche di ambiguità del linguaggio naturale. Infatti, NLP è considerato un arduo problema nell'informatica perché le "leggi" che regolano il linguaggio naturale sono difficili da interpretare per i computer, infatti alcune volte possono sbagliarsi e attribuire significati errati che portano a risultati sbagliati e poco comprensibili. [9]

1.6 Bidirectional Encoder Representations Transformers

Bidirectional Encoder Representations from Transformers (BERT) è un modello di machine learning (ML) basato sui transformers ed utilizzato nell'elaborazione del linguaggio naturale (NLP). In particolare, si tratta di una famiglia di modelli linguistici mascherati introdotta nel 2018 dai ricercatori di Google. Un'indagine sulla letteratura del 2020 ha concluso che "in poco più di un anno, il BERT è diventato una linea di base onnipresente negli esperimenti di NLP contando oltre 150 pubblicazioni di ricerca che analizzano e migliorano il modello". [10][11]

BERT è stato originariamente implementato in lingua inglese in due dimensioni di modello:

- $\text{BERT}_{\text{BASE}}$: composto da 12 codificatori (encoder), ciascuno con 12 teste di auto-attenzione bidirezionali con un totale di 110 milioni di parametri.

- $\text{BERT}_{\text{LARGE}}$: composto da 24 encoder, ciascuno con 16 teste di auto-attenzione bidirezionali con un totale di 340 milioni di parametri.

Entrambi i modelli sono stati pre-addestrati con testo non etichettato quindi in modo non supervisionato, sul Toronto Book Corpus (800 milioni di parole) e su Wikipedia inglese (2.500 milioni di parole). Siccome BERT è un algoritmo e "comprende" i numeri, il testo prima dell'elaborazione deve essere tokenizzato.

In particolare, è stato pre-addestrato su due compiti:

- Modellazione linguistica, 15% dei token era mascherato e l'obiettivo dell'addestramento era prevedere il token originale dato il suo contesto.

- Previsione della frase successiva, l'obiettivo dell'addestramento era stabilire se due sezioni di testo comparivano in sequenza nel corpus di addestramento.

Come risultato di questo addestramento, BERT apprende rappresentazioni latenti di parole e frasi nel contesto. Dopo il "pre-addestramento" o addestramento base di comprensione generale, BERT può essere messo a punto, con meno risorse, su un dataset più piccolo per ottimizzare le sue prestazioni su attività specifiche come NLP e analisi del sentiment. La fase di pre-addestramento è significativamente più onerosa dal punto di vista computazionale rispetto alla messa a punto.

BERT si basa sull'architettura del transformer, in particolare BERT è composto da livelli di encoder transformer. Essendo basato su transformer è in grado di catturare informazioni di contesto da entrambe le direzioni della sequenza di input, ovvero sia da sinistra verso destra che da destra verso sinistra, di conseguenza, acquisisce una profonda comprensione del contesto. [11]

1.6.1 Transformers

Un transformer è un modello di ML basato sull'auto-attenzione, specializzato nell'elaborazione di dati sequenziali. I transformers utilizzano l'auto-attenzione per elaborare sequenze di input come le sequenze di parole in un testo. Il meccanismo dell'auto-attenzione permette al modello di dare maggiore peso a determinate parti dell'input in modo da prestare più attenzione a informazioni rilevanti e ignorare informazioni meno importanti. Il meccanismo di auto-attenzione consente al modello di accedere a tutti gli stati precedenti e valutarli in base a una misura di pertinenza appresa, fornendo informazioni rilevanti sui token lontani. I pesi sono sostanzialmente calcolati negli stati successivi dell'architettura.
Questa capacità di prestare attenzione a parti specifiche del testo rende i transformers particolarmente adatti al NLP dove l'informazione utile può essere dispersa in una sequenza di parole.

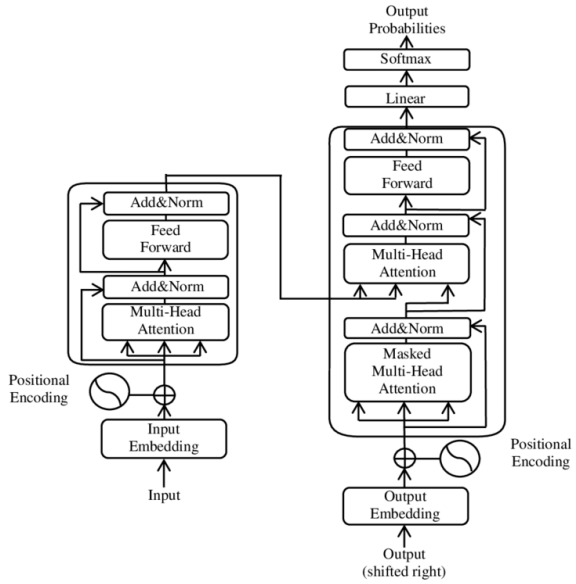

Figure 1: Architettura del trasformatore.

Il trasformatore ha un'elevata parallelizzazione poiché non elabora l'input una parola alla volta in sequenza, ma elabora l'intero input tutto in una volta in modo parallelo, ciò riduce i tempi di addestramento e aumenta l'efficienza in quanto ha una comprensione del testo maggiore.

I transformers utilizzano due meccanismi principali:

- L'attenzione multi-testa, consente al modello di prestare attenzione a più parti specifiche dell'input contemporaneamente.

- L'encoding basato sul trasformatore, permette al modello di catturare informazioni di contesto a lungo termine.

L'architettura del Transformer ha codificatore e decodificatore mentre l'architettura di Transformer usata BERT è leggermente diversa, utilizza solo il codificatore, avendo gli encoder l'uno sull'altro, è quindi composto da livelli di encoder Transformer. [12]

1.7 Overfitting

L'overfitting o sovra adattamento si ha quando un modello statistico molto complesso si adatta ai dati osservati perché ha un numero eccessivo di parametri rispetto ai dati osservati. [13]
Di solito un algoritmo di apprendimento automatico basato sul SL viene addestrato usando un certo insieme di esempi etichettati, si tratta di situazioni tipo di cui è noto il

risultato che si intende prevedere. Dopo la fase di addestramento si assume che il modello predittivo sia in grado di generalizzare, cioè, prevedere l'output dato un input mai visto dal modello. Tuttavia, se il modello è stato addestrato eccessivamente questo non sarà in grado di generalizzare, avrà invece imparato a memoria tutto il training set. Le prestazioni sul training set potranno essere molto buone ma quando si richiede al modello di effettuare predizioni su dati nuovi le performance saranno pessime.
Per evitare l'overfitting, con BERT, il dataset viene diviso in tre parti: training set, evaluation set e test set. Durante l'addestramento il modello viene valutato più volte sul evaluation set per verificare che stia apprendendo nel modo corretto, ovvero che stia generalizzando.

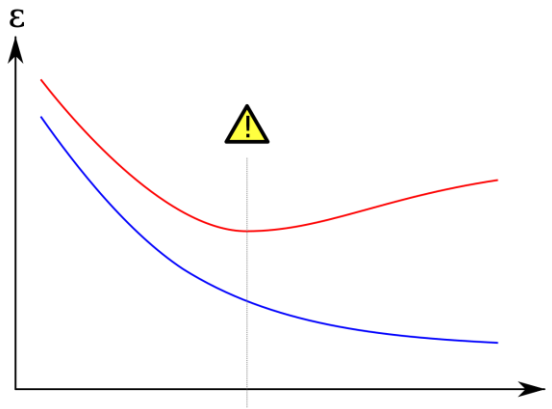

Figure 2: L'eccessivo addestramento porta ad un errore maggiore sul test set.

Curva blu: errore nel classificare i dati di addestramento.
Curva rossa: errore nel classificare i dati di test.

1.8 Sentiment analysis e emotion detection

Sentiment analysis (SA) o opinion mining, è un campo del NLP che basandosi su linguistica computazionale e analisi testuale mira a estrarre ed analizzare documenti provenienti da social media, forum, blog, recensioni, gruppi discussione e qualsiasi altro mezzo attraverso cui possono essere espresse emozioni per rivelare i sentimenti verso prodotti, altre persone o opinioni e trend.
SA ha molte applicazioni, principalmente in statistica, ed è un settore in forte sviluppo, grazie alla sempre maggiore quantità di dati su internet che può essere facilmente strutturata e trasformata in trend ed emozioni espresse verso prodotti, brand, politica e servizi.
Ci sono molti tipi di SA, alcuni si concentrano sulla polarità del testo analizzato e cercando di stabilire se è positiva, negativa o neutrale altri invece si concentrano di più sul capire quali sfumature di emozioni e sentimenti vengono espressi.
Emotion detection (ED) invece si pone l'obiettivo di individuare quali precise emozioni sono presenti nel testo, si tratta quindi più di un problema di classificazione o spotting di

parole chiave. Tipicamente ED fa riferimento a dizionari di emozioni o ad algoritmi di ML come ad esempio il clustering. [14]

1.9 Classificazione delle emozioni

La classificazione delle emozioni è il mezzo con cui si può distinguere un'emozione da un'altra, è un argomento ancora molto discusso e aperto.
I ricercatori principalmente hanno stabilito che le emozioni sono costrutti discreti e fondamentalmente diversi e che possono essere di conseguenza caratterizzate in raggruppamenti.
In questa ricerca è stata usata una delle classificazioni più recenti e dettagliate, quella proposta da Parrott nel 2001 dove ha identificato più di 100 emozioni suddivise in una struttura ad albero con sei emozioni base. In aggiunta alla classificazione di Parrott, per questa ricerca, è stata aggiunta una settima emozione base "neutral" che rappresenta il sentimento neutrale ovvero nessun sentimento. [15]

PARROTT'S EMOTIONS		
PRIMARY EMOTION	**SECONDARY EMOTION**	**TERTIARY EMOTION**
Sadness	Suffering	Agony, Anguish, Hurt
	Sadness	Depression, Despair, Gloom, Glumness, Unhappiness, Grief, Sorrow, Woe, Misery, Melancholy
	Disappointment	Dismay, Displeasure
	Shame	Guilt, Regret, Remorse
	Neglect	Alienation, Defeatism, Dejection, Embarrassment, Homesickness, Humiliation, Insecurity, Insult, Isolation, Loneliness, Rejection
	Sympathy	Pity, Mono no aware, Sympathy
Fear	Horror	Alarm, Shock, Fear, Fright, Horror, Terror, Panic, Hysteria, Mortification
	Nervousness	Anxiety, Suspense, Uneasiness, Apprehension, Worry, Distress, Dread
Anger	Irritability	Aggravation, Agitation, Annoyance, Grouchy, Grumpy, Crosspatch
	Exasperation	Frustration
	Rage	Anger, Outrage, Fury, Wrath, Hostility, Ferocity, Bitterness, Hatred, Scorn, Spite, Vengefulness, Dislike, Resentment
	Disgust	Revulsion, Contempt, Loathing
	Envy	Jealousy
	Torment	Torment
Surprise	Surprise	Amazement, Astonishment
Joy	Cheerfulness	Amusement, Bliss, Gaiety, Glee, Jolliness, Joviality, Joy, Delight, Enjoyment, Gladness, Happiness, Jubilation, Elation, Satisfaction, Ecstasy, Euphoria
	Zest	Enthusiasm, Zeal, Excitement, Thrill, Exhilaration
	Contentment	Pleasure
	Pride	Triumph
	Optimism	Eagerness, Hope
	Enthrallment	Enthrallment, Rapture
	Relief	Relief
Love	Affection	Adoration, Fondness, Liking, Attraction, Caring, Tenderness, Compassion, Sentimentality
	Lust/Sexual desire	Desire, Passion, Infatuation
	Longing	Longing

Figure 3: Classificazione delle emozioni secondo Parrott.

2. Architettura

2.1 Knowledge Discovery in Databases (KDD)

Il termine Knowledge Discovery in Databases, noto anche come KDD, indica l'intero processo di ricerca di nuova conoscenza o informazioni utili da grandi set di dati. Il KDD è un processo in più fasi.
Siccome i dati sono il punto di partenza della ricerca, le prime fasi di preparazione dei dati (come le successive di analisi) sono molto importanti perché vengono preparati i dati su cui viene realizzato lo studio. Se i dati non sono stati preparati nel modo corretto inevitabilmente si avranno risultati incoerenti, errati o fuorvianti quando si applica il modello predittivo, in questo caso per la classificazione automatica delle emozioni.
Il KDD è un processo che consiste in cinque fasi:

- **Selection (selezione)**: Vengono reperiti ed estratti, quindi selezionati, i dati che si intende analizzare dal set di dati. Ovvero, vengono selezionati tutti i post su Stack Overflow dell'anno 2022. I post vengono mantenuti divisi secondo il mese dell'anno.

- **Preprocessing (preelaborazione)**: In questa fase vengono fatte operazioni di base quali la rimozione del rumore e la pulizia dei dati da contenuti non rilevanti per la ricerca. In particolare, vengono rimossi dai post tutti i contenuti che non sono chiaramente testuali e discorsivi come ad esempio codice sorgente, immagini e link.

- **Transformation (trasformazione)**: In questa fase il contenuto dei post viene trasformato, tutte le parole dei post vengono ridotte alla radice per applicare la distant supervision e identificare facilmente tutte le coniugazioni delle parole associate alle emozioni secondo Parrott con l'aggiunta dell'emozione neutrale. Queste etichette verranno usate per l'addestramento di BERT.
  In questa fase viene anche diviso il dataset in training set (60%), evaluation set (20%) e test set (20%).

- **Data mining (estrazione di dati)**: In questa fase ci si occupa dell'estrazione delle informazioni utili dai dati tramite il modello predittivo. Viene applicato BERT sui dati, che effettua la classificazione delle emozioni in modo automatico, dopo essere stato ulteriormente addestrato in modo supervisionato su questo compito specifico.

- **Interpretation and evaluation (interpretazione e valutazione)**: In questa fase vengono valutate le prestazioni del modello, se positive si procede con l'analisi e interpretazione dei risultati. Altrimenti si deve capire se il problema si trova nelle fasi precedenti, dove vengono preparati i dati, oppure se è nell'applicazione non corretta del modello predittivo.

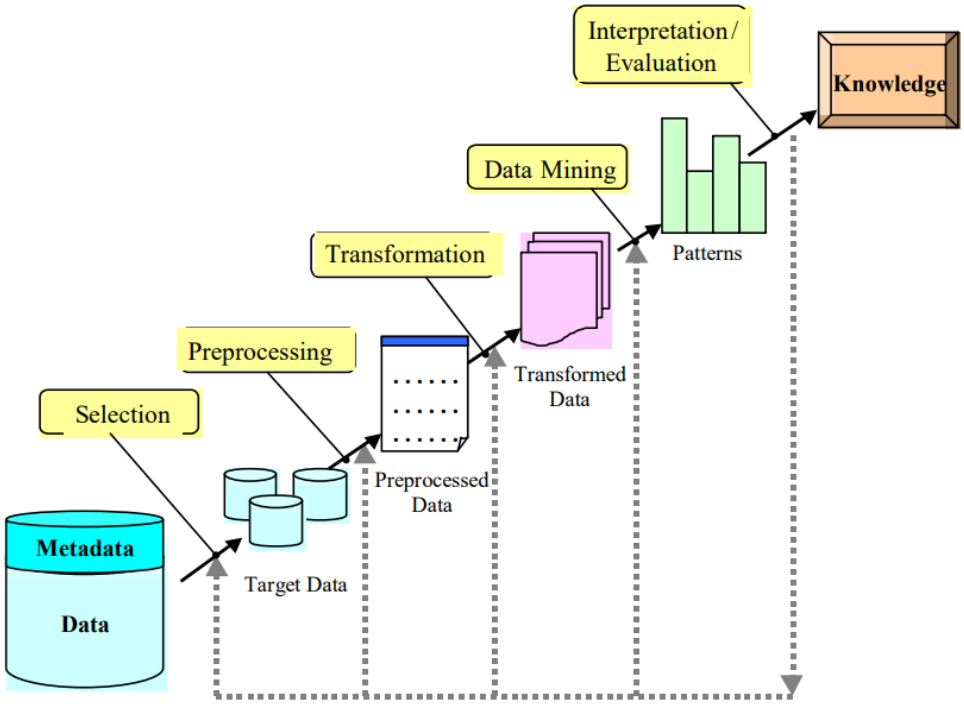

Figure 4: Il processo di KDD.

2.2 Come e dove sono stati reperiti i dati

SE rende periodicamente disponibili i contenuti generati dagli utenti di SO sotto licenza creativa comune. I dati pubblicati sono accessibili in vari modi, è possibile scaricarli come archivi compressi contenenti file XML[16] come è possibile accedervi direttamente online facendo una ricerca selettiva tramite query SQL[17].
Per questo lavoro di ricerca i dati di partenza, ovvero tutti i post su SO nel 2022, sono stati scaricati componendo query SQL. Questa metodologia è stata scelta perché i dati ottenuti con una query vengono già parzialmente organizzati. In particolare, è offerta la possibilità di scaricare i dati in formato CSV (file con caratteri separatori) ed è possibile selezionare i soli attributi di interesse dei post. Questo tipo di formato è relativamente leggero e semplice da gestire rispetto ad una struttura ad albero XML, dove è necessario rimuovere i tag, estrarre le informazioni utili. Inoltre, il file in formato XML contiene tutti i post dal 2008. Bisogna comunque salvare i dati di partenza, dopo aver estratto le informazioni utili, in file CSV perché è un formato molto comodo per fare ML. Tuttavia, c'è un inconveniente, SE impone un limite di 50.000 risultati per query così che i tempi di elaborazione siano ragionevoli per tutti gli utenti. In seguito a questa limitazione, per scaricare tutti i 3.154.807 post del 2022 in modo separato per ogni mese, sono state necessarie 68 query.

2.3 Preprocessing

In questa fase di preprocessing i post vengono analizzati per essere puliti da tutti i contenuti che non sono rilevanti per questo lavoro di ricerca come codice sorgente, immagini, link. Durante questo processo i dati sono stati mantenuti divisi per mese come in tutte le fasi successive. Questo per poter fare un'analisi delle emozioni verso i linguaggi di programmazione nel corso dell'intero anno 2022, cioè per poter osservare i trend.

2.4 Trasformazione dei dati e creazione dei dataset

Nella fase di trasformazione per ogni post vengono analizzate tutte le parole che lo compongono per mettere il formato del testo a minuscolo, rimuovere le parole che non esprimono chiare emozioni per il NLP e ricondurre le parole alla loro radice così da poter applicare la distant supervision in modo semplice. Inoltre, sono stati rimossi i caratteri non stampabili.
Successivamente tutte le istanze (post) vengono etichettate tramite il processo automatico di distant supervision, utilizzando la classificazione gerarchica di Parrott per risalire ad una delle sei emozioni base di Parrott più l'emozione neutrale che viene

assegnata nel caso in cui non è stata trovata nessuna delle emozioni di Parrott oppure se il numero di emozioni positive è uguale al numero di emozioni negative.
Dopo la distant supervision, i post vengono filtrati per mantenere solo quelli che riguardano i linguaggi di programmazione e le tecnologie ad essi associate più popolari nel 2022 secondo il sondaggio annuale di SO. In seguito, i dati vengono mescolati e divisi in 60, 20 e 20% rispettivamente training set, evluation set e test set. Infine, ogni dataset viene perfettamente bilanciato.

2.5 Applicazione del classificatore

Il modello utilizzato per questa ricerca è BERT nella dimensione base con 12 encoder e 110 milioni di parametri, i parametri sono impostati in modo automatico dal modello durante la fase di addestramento e ottimizzati, sempre dal modello, nella sottofase di valutazione dell'addestramento. Il modello viene addestrato per associare ad ogni istanza una delle sette classi quali *neutral*, *love*, *joy*, *surprise*, *anger*, *fear* e *sadness*. Oltre all'addestramento viene fatta anche la validazione per evitare che il modello impari il training set a memoria invece che generalizzare, ovvero per evitare l'overfitting. La fase di validazione, che viene effettuata durante la fase di addestramento, serve anche per ottimizzare i pesi dei parametri scelti dall'algoritmo nella fase di addestramento. Successivamente, per ottenere le predizioni viene applicato il modello sul test set, che è composto da dati mai visti dal modello, e vengono misurate le prestazioni. Infine, il modello addestrato viene salvato ed applicato su tutti i post di SO riguardanti le tecnologie più popolari secondo Stack Overflow Developer Survey 2022[18].

3. Implementazione

3.1 Approfondimento sulla selezione dei dati

SE dà la possibilità di effettuare query SQL ad un dump del database di Stack Overflow qui https://data.stackexchange.com/stackoverflow/query/new. I dati relativi ai post del 2022 sono stati reperiti dalla tabella "Posts" con schema:

Posts(Id, PostTypeId, AcceptedAnswerId, ParentId, CreationDate, DeletionDate, Score, ViewCount, Body, OwnerUserId, OwnerDisplayName, LastEditorUserId, LastEditorDisplayName, LastEditDate, LastActivityDate, Title, Tags, AnswerCount, CommentCount, FavoriteCount, ClosedDate, ComunityOwnedDate, ContentLicense).

Questa tabella contiene tutte le domande e risposte pubblicate dagli utenti dal 2008 al momento dell'ultimo dump che è stato nel 2023, il contenuto viene aggiornato periodicamente. Con una query count si trova che il numero di post pubblicati nel 2022 è 3.154.807, questo numero può variare perché i post possono essere rimossi da SE, infatti negli anni precedenti alcuni milioni di domande senza risposta, considerate archiviate, sono state eliminate.

Gli attributi più importanti sono:

- **Id**: indica un id univoco per ogni istanza.
- **PostTypeId**: indica il tipo di post, se ha valore 1 significa che è una domanda invece se ha valore 2 significa che è una risposta. Può avere anche altri valori ma non sono rilevanti per questa ricerca.
- **AcceptedAnswerId**: indica l'id della risposta accettata dal creatore della domanda se il PostTypeId è uguale a 1.
- **ParentId**: indica l'id della domanda se il PostTypeId è uguale a 2.
- **CreationDate**: indica la data di creazione del post.
- **Body**: indica la domanda/risposta di un utente in formato HTML.
- **Title**: indica il titolo del post.

- **Tags**: indica i tag inseriti dal creatore del post o aggiunti da un altro utente. I tag sono presenti solo nei post di tipo 1.

- **AnswerCount**: indica il numero di risposte ricevute.

- **CommentCount**: indica il numero di commenti ricevuti.

Per questo lavoro sono stati selezionati inizialmente Body, Title e Tags. Il Body ovviamente perché contiene il post che si vuole analizzare. Il Title poteva essere utile per una prima analisi ma successivamente è stato scartato perché contiene sostanzialmente solo quale è l'argomento tecnico del post. I Tags, come il Body, invece sono essenziali per questo lavoro di ricerca poiché vengono utilizzati per filtrare il dataset ai soli linguaggi di programmazione e tecnologie associate secondo la classifica annuale di SO per il 2022 (Stack Overflow Developer Survey 2022[18]).
Dato che, SE impone un limite di 50.000 risultati alle query, in totale sono state necessarie 68 query SQL per ottenere tutti i post del 2022 su SO. Questo limite viene imposto per garantire un tempo di risposta ragionevole per tutti.

3.2 Organizzazione dei dati e preprocessing

In questa fase i post vengono scaricati e ordinati per mese in dodici cartelle, ognuna delle quali contiene i post del corrispettivo mese. Per via del limite di 50.000 risultati per query, i post di ogni mese hanno richiesto in media 6 file. I dati sono stati scaricati in formato CSV. Successivamente con uno script Python sono stati analizzati tutti i post per una pulizia preliminare in modo da rimuovere il rumore, e per ogni mese è stato creato un file unico che contiene i post quasi completamente puliti dal rumore e informazioni non rilevanti per questa ricerca. In particolare, tramite la libreria *re* usata per applicare le espressioni regolari in Python sono stati rimossi dal Body, che inizialmente era in formato HTML, codice sorgente, immagini, link e caratteri non stampabili. Alla fine di questo processo si ottiene un body, per ogni post, quasi completamente composto dal contenuto testuale che si vuole analizzare, ovvero la domanda dell'utente.

Numero iniziale di istanze per ogni mese:

Mese	**Istanze**
Gennaio	225.973
Febbraio	261.142
Marzo	280.710

Aprile	246.303
Maggio	292.495
Giungo	282.825
Luglio	205.130
Agosto	288.803
Settembre	267.985
Ottobre	272.676
Novembre	282.420
Dicembre	248.345
Totale	3.154.807

3.3 Approfondimento sulla trasformazione dei dati e distant supervision

3.3.1 Trasformazione dei dati

Per poter applicare la distant supervision in modo semplice ed efficace, sfruttando la grande mole di dati, è necessario fare delle semplici trasformazioni dei post. Questo per poter riconoscere con facilità le emozioni base di Parrott e tutti i loro sottolivelli. La suddivisione di Parrott delle emozioni coinvolge più di cento parole ognuna associata ad un'emozione. Siccome ogni parola può avere molte coniugazioni o sinonimi non è pratico avere un dizionario con tutte le possibili coniugazioni o sinonimi della parola per poterla identificare nel testo. Dunque, quello che viene fatto è ricondurre le parole alla loro radice, così da avere un dizionario delle parole da identificare molto più gestibile, questo rende anche l'analisi delle parole in un post più semplice dal punto di vista computazionale. Il post viene mantenuto anche in forma originale per la successiva analisi con il modello predittivo BERT.

Con uno script Python, sia per il post che viene mantenuto in forma originale che per quello che viene usato nella distant supervision, vengono completate le operazioni di pulizia rimuovendo i caratteri non stampabili poi viene messo il testo minuscolo per evitare testi completamente maiuscoli. Invece solo per i post copia, utilizzati

successivamente per la distant supervision tramite la libreria *nltk*, vengono ricondotte le parole alla loro radice e rimosse le parole chiaramente prive di emozioni come gli articoli. I post vengono prelevati ed elaborati uno alla volta, da file CSV, così da avere nello stesso istante il post originale e la copia per la distant supervision, in questo modo dopo la preparazione vengono riscritte nella stessa riga (in un file diverso) il post originale e la copia. Questo ordinamento viene usato perché in fase di distant supervision, dove i post vengono elaborati in modo altrettanto sequenziale, si possa associare l'etichetta definita per la copia del post preparata al post originale. Nel file di output quindi si mantengono post originale, post trasformato per la distant supervision e i tags. L'attributo titolo è stato scartato perché non utile per l'analisi, come notato nel paragrafo 3.1. I tags sono un attributo mantenuto e non utilizzato per le fasi iniziali fino al filtraggio. I tags vengono anche utilizzati alla fine per l'interpretazione dei risultati dove per ogni linguaggio di programmazione si guarda il sentiment. I tags sono utilizzati per individuare i linguaggi di programmazione.

3.3.2 Distant supervision (algoritmo utilizzato)

La suddivisione delle emozioni gerarchica di Parrott è stata passata da tre livelli a due. Il terzo e il secondo sono stati uniti in uno unico per ogni emozione, così da mantenerli in una sola lista. Per ognuna delle sei emozioni base di Parrott più l'emozione neutrale viene mantenuto un contatore, questo per trovare la parola associata all'emozione più frequente. Successivamente si guarda per ogni parola in ogni post se questa sta nel dizionario a due livelli, ovvero prima si verifica se la parola corrisponde a una delle sei associate alle emozioni base, se corrisponde si incrementa il corrispettivo contatore, se non corrisponde si cerca la parola nel secondo livello, se si ha una corrispondenza allora viene incrementato il contatore dell'emozione di primo livello corrispondente e viene salvata la parola che ha causato l'incremento. Invece se in tutto il processo non è mai stata identificata un'emozione oppure se il numero di emozioni positive è uguale al numero di emozioni negative viene direttamente associata l'etichetta neutrale. Dopo aver analizzato tutte le parole di un post si definisce l'etichetta come quella con il contatore massimo, identificata l'etichetta si cerca la parola, tra quelle appositamente salvate, che ha causato la scelta di tale etichetta e la si rimuove dal post originale per evitare che il modello impari l'associazione parola-etichetta, altrimenti il modello ogni volta che individua la parola associa direttamente l'etichetta. Ciò porta ad un risultato falsato e non veritiero.

L'intero processo viene fatto con uno script Python. I post vengono letti ed elaborati uno alla volta dal file CSV prodotto nella fase precedente di trasformazione. In questo modo per ogni post che si analizza si ha sempre la coppia (post originale, post trasformato). Per stabilire l'etichetta viene analizzato il post trasformato, stabilita l'etichetta si salva in un nuovo file il post originale, l'etichetta e ovviamente i tags necessari per l'interpretazione finale dei risultati.

Di seguito una tabella esplicativa sulla classificazione delle emozioni. Le emozioni, come le classi, sono in inglese perché i post su Stack Overflow, cioè i testi analizzati sono ovviamente in inglese.

Classificazione gerarchica delle emozioni a due livelli	
Emozione base	**Secondo e terzo livello**
love	Affection, adoration, fondness, attraction, caring, tenderness, compassion, sentimentality, lust, arousal, desire, passion, infatuation
joy	Cheerfulness, amusement, bliss, gaiety, glee, jolliness, joviality, delight, enjoyment, gladness, happiness, jubilation, elation, satisfaction, ecstasy, euphoria, zest, enthusiasm, zeal, excitement, thrill, exhilaration, contentment, pleasure, pride, triumph, optimism, eagerness, hope, enthrallment, rapture, relief
surprise	Amazement, astonishment
fear	Nervousness, anxiety, tenseness, uneasiness, apprehension, worry, distress, dread, horror, alarm, shock, fright, terror, panic, hysteria, mortification
sadness	Suffering, agony, hurt, anguish, depression, despair, hopelessness, gloom, glumness, unhappiness, grief, sorrow, woe, misery, melancholy, disappointment, dismay, displeasure, shame, guilt, regret, remorse, neglect, alienation, isolation, loneliness, rejection, homesickness, defeat, dejection, insecurity, embarassment, humiliation, insult, sympathy, pity
anger	Irritation, aggravation, agitation, annoyance, grouchiness, grumpiness, exasperation, frustration, rage, outrage, fury, wrath, hostility, ferocity, bitterness, hate, loathing, scorn, spite, vengefulness, dislike, resentment, disgust, revulsion, contempt, envy, jealousy, torment
neutral	None of the above identified

3.4 Approfondimento sulla creazione dei dataset

3.4.1 Filtraggio dei linguaggi di programmazione

Siccome questo lavoro di ricerca mira ad osservare il sentiment verso i più popolari linguaggi di programmazione e tecnologie ad essi associate, che sono ben 42! secondo Stack Overflow Developer Survey 2022[18], è necessario filtrare i post e preservare soltanto quelli che rientrano nella classifica di SO. Per questa operazione si utilizza l'attributo tags.

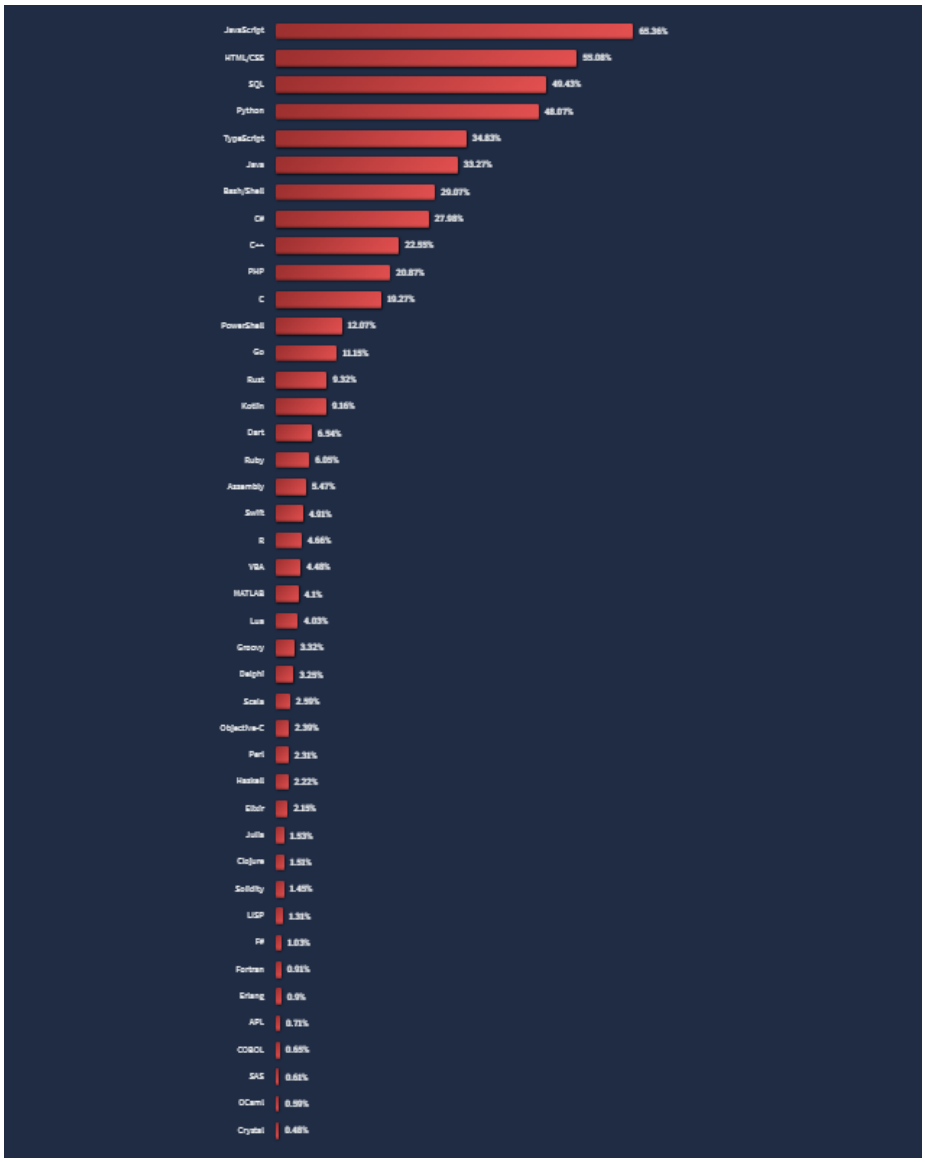

Figure 5: Tecnologie più popolari secondo Stack Overflow Developer Survey 2022.

Con uno script Python si analizzano i file CSV contenenti i post. Per ogni post vengono guardati i tags e se c'è almeno un tag tra uno dei 42 allora il post viene mantenuto per le fasi successive, e in particolare per l'analisi finale, altrimenti viene scartato.

Numero di istanze per ogni mese dopo il filtraggio:

Mese	**Istanze**
Gennaio	62.259
Febbraio	72.283
Marzo	78.660
Aprile	74.110
Maggio	86.709
Giungo	82.525
Luglio	59.400
Agosto	82.480
Settembre	76.464
Ottobre	80.325
Novembre	83.600
Dicembre	71.926
Totale	827.141

3.4.2 Divisione del dataset

In questa fase il dataset viene diviso in tre parti, ovvero training set, evaluation set, test set rispettivamente con percentuali 60%, 20% e 20%. Il training set viene utilizzato per l'addestramento, sul task specifico di classificazione del testo, del modello BERT. Mentre evaluation set viene utilizzato sempre durante la fase di addestramento per ridurre il rischio di overfitting. Infine, il test set è ovviamente utilizzato alla fine dell'addestramento per far fare predizioni al modello su dati mai visti prima.
Fino ad ora il dataset è stato mantenuto in dodici pezzi, ovvero dodici cartelle una per ogni mese, ognuna delle quali contiene un file creato per ogni fase. Questo perché non è possibile leggere un file calcolare l'output e scrivere il risultato nello stesso file e per

mantenere la divisione mensile dei dati. Ogni fase produce un file aggiornato per la fase successiva.

L'operazione di creazione dei tre dataset viene fatta con uno script Python. Per ogni mese, prima di dividere il file nelle tre componenti secondo le percentuali indicate sopra, il contenuto viene mescolato in modo pseudo casuale per mantenere la massima generalità possibile. Il training set viene realizzato mettendo insieme il 60% del file contenente i post per ogni mese. Nello stesso modo viene realizzato evaluation set, però utilizzando il 20% del file. Il test set viene realizzato con il restante 20%, però non viene unito a differenza degli altri viene mantenuto separato. Il test set è quindi composto da dodici file, uno per mese, così da non avere necessità di salvare la data del post in quanto la divisione stessa dei file in dodici definisce la data. Tale metodologia abbassa anche la complessità perché non si deve controllare la data per l'analisi finale. È importante mantenere i post di test, cioè i post su cui il modello effettua le predizioni, separati per mese così da poter osservare i trend del sentiment lungo il corso dell'anno 2022.

3.4.3 Bilanciamento dei dataset

Per un corretto addestramento del modello predittivo BERT è necessario che il dataset sia bilanciato, in particolare in questa fase viene perfettamente bilanciato. Questa operazione è importante perché se ci fossero tante istanze di una emozione e meno delle altre, il modello imparerebbe che è più probabile ci sia l'emozione più ripetuta concentrandosi troppo su di essa.

Numero di istanze individuate per ogni emozione in ciascun dataset:

Training set

Emozione	**Istanze**
neutral	506.570
love	14.453
joy	17.177
surprise	1.280
anger	1.675
sadness	2.800
fear	2.487

Evaluation set

Emozione	Istanze
neutral	168.911
love	4.749
joy	5.843
surprise	398
anger	530
sadness	934
fear	780

Test set gennaio

Emozione	Istanze
neutral	11.535
love	334
joy	406
surprise	31
anger	36
sadness	62
fear	49

Test set febbraio

Emozione	Istanze
neutral	13.398
love	378
joy	460
surprise	34
anger	45
sadness	80
fear	63

Test set marzo

Emozione	**Istanze**
neutral	14.576
love	404
joy	503
surprise	37
anger	48
sadness	95
fear	69

Test set aprile

Emozione	**Istanze**
neutral	13.736
love	384
joy	469
surprise	35
anger	47
sadness	77
fear	74

Test set maggio

Emozione	**Istanze**
neutral	16.133
love	419
joy	515
surprise	48
anger	53
sadness	91
fear	84

Test set giugno

Emozione	**Istanze**
neutral	15.305
love	422
joy	531
surprise	35
anger	51
sadness	95
fear	66

Test set luglio

Emozione	**Istanze**
neutral	11.028
love	327
joy	360
surprise	29
anger	31
sadness	55
fear	50

Test set agosto

Emozione	**Istanze**
neutral	15.307
love	407
joy	510
surprise	51
anger	45
sadness	89

fear	87

Test set settembre

Emozione	**Istanze**
neutral	14.227
love	417
joy	417
surprise	31
anger	42
sadness	77
fear	83

Test set ottobre

Emozione	**Istanze**
neutral	14.925
love	426
joy	480
surprise	36
anger	43
sadness	85
fear	70

Test set novembre

Emozione	**Istanze**
neutral	15.479
love	443
joy	545
surprise	33
anger	50

sadness	93
fear	77

Test set dicembre

Emozione	**Istanze**
neutral	13279
love	387
joy	469
surprise	41
anger	45
sadness	92
fear	73

A questo punto è semplice capire perché sia necessario un bilanciamento.

Il bilanciamento perfetto viene realizzato tramite uno script Python. Come prima cosa viene individuato il numero minimo di istanze per ogni dataset. Per training set è 1.280, per evaluation set è 398 e per il test set è 29. A questo punto per ogni dataset si scorre il corrispettivo file CSV copiando in un altro file CSV le istanze fino a raggiungere il minimo assoluto individuato precedentemente, per ogni classe si ottiene tale numero di istanze. Dopo il bilanciamento il numero di istanze complessivo per training set è 7*1.280=8.960, per evaluation set è 7*398=2.786 e per test set è 7*29*12=2.436.

3.5 Approfondimento sul classificatore

Per creare il modello predittivo viene utilizzata la classe *BertForSequenceClassification* della libreria *transformers* di Hugging Face. In particolare, la classe viene creata facendo riferimento al modello base BERT pre-addestrato in modo non supervisionato (*bert-base-uncased*). Questa classe è stata scelta perché è appositamente progettata per la classificazione del testo. La classe aggiunge un ulteriore livello alla rete neurale BERT per la classificazione, di conseguenza il modello deve essere addestrato sul task specifico di classificazione prima di poter essere utilizzato. In particolare, vengono aggiunte sette teste di output ognuna delle quali dà una probabilità di appartenenza dell'esempio considerato alla classe rappresentata dalla testa di output, la probabilità più alta è la predizione del modello. Il numero di classi viene passato come argomento insieme al dizionario etichetta-numero corrispondente (label2id) e il dizionario inverso (id2label) per la creazione della classe *BertForSequenceClassification.* BERT "capisce" solo i numeri per questo bisogna scegliere un numero corrispondente ad ogni etichetta e passargli il dizionario come argomento. Anche per i dataset è necessario convertire le etichette in numeri, così come si devono tokenizzare i post. Per la tokenizzazione viene utilizzata la classe *BertTokenizer* della libreria *transformers*, sempre basata sul modello *bert-base-uncased*.
Con uno script Python viene creato ed addestrato il modello, successivamente vengono effettuate e valutate le predizioni.

```
id2label = {0: "neutral", 1: "love", 2: "joy", 3: "surprise", 4: "anger", 5: "sadness", 6: "fear"}
label2id = {"neutral": 0, "love": 1, "joy": 2, "surprise": 3, "anger": 4, "sadness": 5, "fear": 6}

model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=7, id2label=id2label, label2id=label2id)

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
```

Figure 6: Definizione di dizionari, modello e tokenizer.

3.5.1 Rappresentazione del dataset

BERT della libreria *transformers* necessita di una classe che rappresenta il dataset. La classe deve contenere come attributi la lista dei post tokenizzati (input_ids), le etichette ovviamente in forma numerica corrispondenti nel caso dell'addestramento (labels) e la lista degli indici che specificano quali token devono essere presi in considerazione dal modello (attention_mask). Per generare gli input_ids (tokenizzare i post) e generare attention_mask viene usata la classe *BertTokenizer*.
Questa classe deve avere almeno i metodi *__len__* e *__getitem__*, usati da BERT per comprendere il testo da analizzare. Il metodo *__len__* deve restituire la lunghezza della lista input_ids, mentre il metodo *__getitem__* deve restituire la lista input_ids e attention_mask più opzionalmente le etichette nel caso la classe stia rappresentando i

dati per l'addestramento. Sostanzialmente il metodo restituisce gli elementi da analizzare in un formato comprensibile per BERT.

```
class SimpleDataset:
    def __init__(self, input_ids, attention_masks, labels=None):
        self.input_ids = input_ids
        self.attention_masks = attention_masks
        self.labels = labels

    def __len__(self):
        return len(self.input_ids)

    def __getitem__(self, idx):
        if self.labels is not None:
            return {'input_ids': self.input_ids[idx], 'attention_mask': self.attention_masks[idx], 'labels': self.labels[idx]}
        else:
            return {'input_ids': self.input_ids[idx], 'attention_mask': self.attention_masks[idx]}
```

Figure 7: Classe SimpleDataset, rappresenta un dataset generico.

3.5.2 Preparazione dei dati

Siccome i dataset sono mantenuti in formato CSV vengono utilizzate le librerie *pandas* e *csv* per leggere i dataset e convertire ogni colonna del file di partenza in una lista. Nel caso della preparazione dei dati di addestramento vengono letti e salvati in lista i post e le etichette. Le etichette vengono convertite in numeri utilizzando il dizionario appositamente creato label2id. Successivamente viene utilizzato il tokenizer per tokenizzare i post da analizzare, come argomenti del tokenizer oltre a passare i post viene abilitato il troncamento e il padding così che il testo analizzato dal modello non sia troppo lungo e abbia sempre la stessa lunghezza, come lunghezza massima del post tokenizzato viene passato 512 in modo tale che ci stiano praticamente tutti i post. Infine, si crea un'istanza della classe SimpleDataset passando input_ids, attention_mask del testo tokenizzato e le etichette. Questa istanza rappresenta il training set. Il procedimento è analogo per evaluation set mentre per i dodici test set non vengono passate le etichette ovviamente.

```
train_dataset_csv = pd.read_csv(TRAIN, encoding='utf-8', sep=';')
train_texts = train_dataset_csv['post'].dropna().astype('str').tolist()
train_labels = train_dataset_csv['label'].dropna().astype('str').tolist()
train_labels_int = [label2id[label] for label in train_labels]
tokenized_traintexts = tokenizer(train_texts,truncation=True,padding=True,max_length=512)
train_dataset = SimpleDataset(tokenized_traintexts['input_ids'],tokenized_traintexts['attention_mask'],train_labels_int)
```

Figure 8: Preparazione del training set.

3.5.3 Addestramento e predizioni del modello

BERT ha già un addestramento base realizzato in modo non supervisionato, ma questo gli permette soltanto di avere una comprensione generale e profonda del testo. BERT per

poter essere implementato con successo per un compito specifico deve essere ulteriormente addestrato in modo supervisionato.
Per addestrare il modello vengono utilizzate le classi *TrainingArguments* e *Trainer*, rispettivamente per definire gli argomenti utilizzati in fase di addestramento e per addestrare il modello e effettuare predizioni.

Argomenti di addestramento utilizzati:

- **output_dir**: indica la cartella in cui viene salvato il modello addestrato (i checkpoint secondo la strategia di salvataggio) e le predizioni.

- **evaluation_strategy**: indica la strategia di valutazione del modello, per questa ricerca viene utilizzato il valore "epoch" cioè il modello viene valutato sul evaluation set ogni epoca di addestramento.

- **learning_rate**: indica il tasso di apprendimento iniziale per l'addestramento, il valore utilizzato è "2e-5", si tratta di un valore tipo per task di NLP, l'intervallo per dataset di medie dimensioni è [1e-5;5e-5]. Il modello in fase di addestramento può cambiare il valore per ottimizzarlo. Questo valore è tipicamente utilizzato perché dà buoni risultati nella maggior parte dei casi.

- **per_device_train_batch_size**: indica il numero di esempi elaborati contemporaneamente su ogni CPU/GPU durante l'addestramento. Il valore utilizzato è "16".

- **per_device_eval_batch_size**: indica il numero di esempi elaborati contemporaneamente su ogni CPU/GPU durante la valutazione del modello che è fatta in fase di addestramento. Il valore utilizzato è "16".

- **num_train_epochs**: indica il numero di volte che il modello viene addestrato su tutto il training set, per evitare l'overfitting non deve essere alto. Il valore utilizzato è "3".

- **weight_decay**: indica il decadimento del peso, il valore impostato è un termine di penalità che viene aggiunto alla funzione di costo della rete neurale che ha l'effetto di ridurre l'overfitting in quanto se alto limita l'eccessiva crescita dei pesi. Il decadimento del peso è una tecnica di regolarizzazione nota anche come L2. Il valore utilizzato è "0.01" che significa una regolarizzazione moderata, l'intervallo di utilizzo tipico è [0.1; 0.001].

- **push_to_hub**: indica se caricare il modello addestrato su Hugging Face. Il valore utilizzato è "False".

- **logging_dir**: indica la cartella in cui vengono salvati i log di addestramento del modello.

- **save_strategy**: indica la strategia di salvataggio del modello addestrato, ovvero ogni quanto salvare un checkpoint. Il valore utilizzato è "epoch", ad ogni epoca di addestramento viene salvato un checkpoint in **output_dir**. In questo caso vengono salvati tre checkpoint perché il numero di epoche è tre.

- **load_best_model_at_end**: indica se caricare il modello con le prestazioni migliori alla fine dell'addestramento. Il valore utilizzato è "True", questo significa che alla fine dell'addestramento viene utilizzato come modello quello che ha ottenuto le prestazioni migliori in fase di valutazione.

```
training_args = TrainingArguments(
    output_dir='./BERTdata',
    evaluation_strategy = "epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=3,
    weight_decay=0.01,
    push_to_hub=False,
    logging_dir='./BERTlogs',
    save_strategy='epoch',
    load_best_model_at_end=True,
)
```

Figure 9: Argomenti di addestramento.

Per implementare effettivamente il modello è necessario creare un'istanza della classe *Trainer*. L'istanza creata rappresenta il modello passatogli come argomento insieme ai dataset di addestramento e valutazione. Per addestrare il modello si utilizza il metodo *train* senza passare argomenti mentre per fare una predizione si utilizza il metodo *predict* con argomento il test set. Il test set deve essere preparato come visto nel paragrafo 3.5.2.

Argomenti del Trainer:

- **model**: indica il modello pre-addestrato da utilizzare. Viene utilizzato *BertForSequenceClassification* basato su *bert-base-uncased*.

- **args**: indica gli argomenti di addestramento, deve essere un oggetto della classe *TrainingArguments* vista sopra.

- **train_dataset**: indica il training set che viene utilizzato per l'addestramento del modello. Il training set deve essere preparato come mostrato nel paragrafo 3.5.2.

- **eval_dataset**: indica evaluation set che viene utilizzato in fase di addestramento per la valutazione del modello, in particolare per ottimizzare i pesi e verificare che

il modello stia generalizzando bene. Evaluation set deve essere preparato come mostrato nel paragrafo 3.5.2.

- **tokenizer**: indica il tokenizer utilizzato per tokenizzare i dati che il modello deve analizzare. Viene utilizzato *BertTokenizer* basato su *bert-base-uncased*, come definito all'inizio della sezione 3.5.

```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    tokenizer=tokenizer
)

trainer.train()
```

Figure 10: Definizione del Trainer e addestramento del modello.

Le predizioni del modello, essendo numeriche, vengono convertite in etichette tramite il dizionario id2label definito all'inizio della sezione 3.5. Le predizioni del modello vengono salvate in file CSV.

3.5.4 Prestazioni del modello

Misurare le prestazioni del modello, ovvero la correttezza delle predizioni, è un'attività molto importante per capire se le predizioni del modello sono buone. Se le prestazioni non sono buone può dipendere da come sono stati preparati i dati in precedenza oppure da una scorretta impostazione del modello.
Tramite il modello definito nei paragrafi precedenti vengono fatte dodici predizioni su dodici test set diversi mai visti prima dal modello, uno per ogni mese dell'anno 2022. Per ognuna di queste dodici predizioni vengono misurate le prestazioni utilizzando le funzioni *accuracy_score* e *precision_recall_fscore_support* della libreria *sklearn.metrics*. Con la prima funzione si misura l'accuratezza del modello, che indica la percentuale di istanze del test set classificate correttamente. Con la seconda funzione si misurano la precisione, la sensibilità (recall) e il punteggio F1 (F1-score). La precisione misura la percentuale di esempi positivi correttamente identificati dal modello rispetto al totale di esempi identificati come positivi dal modello. Il recall misura la capacità del modello di identificare correttamente tutti gli esempi positivi di una classe rispetto al totale di esempi positivi presenti nel dataset. Infine, F1-score tiene conto sia della precisione che del recall, misura la capacità bilanciata del modello di identificare correttamente sia gli esempi positivi che quelli negativi di una classe.

Prestazioni del modello in percentuale per ognuna delle dodici predizioni:

Mese/test set	**Accuratezza**	**Precisione**	**Recall**	**F1-score**
Gennaio	0,6896	0,6948	0,6896	0,6870
Febbraio	0,6798	0,6957	0,6798	0,6857
Marzo	0,6748	0,6853	0,6748	0,6765
Aprile	0,6157	0,6282	0,6157	0,6147
Maggio	0,6403	0,6456	0,6403	0,6411
Giugno	0,7142	0,7272	0,7142	0,7184
Luglio	0,6256	0,6623	0,6256	0,6358
Agosto	0,6453	0,6754	0,6453	0,6553
Settembre	0,6600	0,6644	0,6600	0,6609
Ottobre	0,6256	0,6394	0,6256	0,6261
Novembre	0,7142	0,7102	0,7142	0,7101
Dicembre	0,6453	0,6443	0,6453	0,6422

L'accuratezza media del modello è del 66% nell'individuare a quale delle 7 classi appartiene un esempio, che è un buon risultato poiché individuare una classe tra sette non è facile.
Successivamente un'ulteriore misurazione delle prestazioni è stata fatta per individuare l'accuratezza media nello stabilire la polarità del testo analizzato. Per ottenere questa misura le etichette, per tutti i dodici test set, sono state sostituite con 'pos' se rappresentati emozioni positive (*love*, *joy*, *surprise*) e con 'neg' se rappresentanti emozioni negative (*anger*, *fear*, *sadness*). Mentre tutte le istanze con etichetta *neutral* definita in fase di distant supervision sono state rimosse perché non significative. Si trova che l'accuratezza media nello stabilire la polarità del testo è del 77%, poiché in questo caso bisogna scegliere tra sole due classi, si ha un significativo miglioramento delle prestazioni.

3.5.5 Applicazione del modello a SO (Big data analytics)

Ora che il modello (BERT) è stato addestrato e salvato nella cartella *output_dir* di *TrainingArguments* durante la fase precedente, in quest'ultima fase di implementazione viene applicato a tutte le domande su SO riguardanti le 42 tecnologie più popolari individuate in Stack Overflow Developer Survey 2022. In particolare, il modello viene applicato alle 827.141 istanze, divise nei dodici mesi dell'anno 2022, rimanenti dopo il filtraggio delle tecnologie (paragrafo 3.4.1).
Il modello è stato addestrato con tre epoche di addestramento e la strategia di salvataggio che è stata utilizzata è ad epoche, di conseguenza sono stati salvati tre checkpoint o versioni del modello addestrato nella cartella *output_dir*. La versione del modello utilizzata per classificare tutte le domande è l'ultima perché durante l'addestramento, nella sottofase di valutazione, ha dato le migliori performance anche perché il modello è stato addestrato di più che nei checkpoint precedenti.

Il modello viene definito come visto all'inizio della sezione 3.5 con la differenza che al posto di *'bert-base-uncased'* viene messo il percorso della cartella contenente il terzo checkpoint del modello precedentemente addestrato. Il *tokenizer* rimane invariato. Il dataset, diviso in dodici parti una per ogni mese, deve essere preparato come visto nel paragrafo 3.5.2. In questo caso gli argomenti di addestramento, il training set ed evaluation set non sono necessari perché il modello è stato addestrato in precedenza. Di conseguenza, il trainer viene definito passando solo il modello e il *tokenier*, necessario perché viene utilizzato per effettuare le predizioni tramite il metodo *predict* con parametro il dataset preparato. Le predizioni vengono convertite tramite il dizionario da numeri a etichette e salvate in file CSV.
I risultati ottenuti dal modello in questa fase vengono analizzati nel capitolo successivo.

4. Risultati

Come indicato nel paragrafo 3.4.1 il dataset iniziale è stato filtrato per mantenere soltanto i post riguardanti le 42 tecnologie più popolari secondo Stack Overflow Developer Survey 2022[18], si è quindi passati dalle 3.154.807 istanze iniziali alle 827.141. Il numero di istanze è notevolmente diminuito anche perché le risposte non hanno tags, di conseguenza è complesso assegnare la tecnologia a cui appartengono.

Il dataset utilizzato per la Big data analytics è diviso in dodici parti, una per ogni mese dell'anno 2022, così da poter osservare come variano le emozioni espresse verso i linguaggi di programmazione e tecnologie ad essi associate predette dal modello.

4.1 Tecnologie più popolari secondo il modello

Nella statistica di questo paragrafo vengono scartate le emozioni della classe *neutral*, così da avere esattamente tre classi di emozioni positive (*love*, *joy*, *surprise*) e tre di emozioni negative (*anger*, *sadness*, *fear*). Questo perché l'emozione neutrale non influisce in una statistica di valutazione positiva e valutazione negativa.
Per realizzare la classifica seguente dei linguaggi di programmazione e tecnologie ad essi associate più popolari vengono considerate le etichette predette dal modello. In particolare, per ogni tecnologia si contano le istanze per ognuna delle tre emozioni positive e negative, successivamente si calcola il totale delle emozioni positive e di quelle negative, infine si riporta la percentuale di emozioni positive e quelle negative. Per la seguente statistica le tecnologie con meno di 1.000 istanze totali tra emozioni positive e negative non vengono riportate.

Di seguito la percentuale delle tecnologie maggiormente amate e odiate più il numero di emozioni positive e negative per ogni tecnologia individuata, con almeno 1.000 istanze totali, nel test set tra i 42 di Stack Overflow Developer Survey 2022. La tabella è ordinata dalla più alla meno apprezzata.

Tecnologia	**Positive (%)**	**Negative (%)**	**Tot. Positive**	**Tot. Negative**
MATLAB	77,95	22,05	859	243
R	77,68	22,32	19.243	5.529
SQL	72,79	27,21	10.328	3.861
Python	64,7	35,3	67.753	36.960

Assembly	63,59	36,41	793	454
C	59,67	40,33	5.841	3.948
Shell	58,49	41,51	1.275	905
Bash	58,25	41,75	2.571	1.843
Scala	57,62	42,38	1.006	740
CSS	57,33	42,67	7.754	5.772
VBA	53,85	46,15	2.440	2.091
C++	53,68	46,32	10.081	8.700
HTML	51,03	48,97	9.296	8.922
PowerShell	48,23	51,77	2.451	2.631
Go	46,33	53,67	1.738	2.013
JavaScript	45,82	54,18	26.289	31.089
Rust	43,89	59,11	1.777	2.272
PHP	42,49	57,51	6.289	8.511
Ruby	42,35	57,65	947	1.289
Java	42,21	57,79	13.486	18.465
TypeScript	40,29	59,71	5.614	8.320
C#	39,85	60,15	11.154	16.834
Swift	39,7	60,3	3.083	4.682
Kotlin	38,78	61,22	2.588	4.085
Dart	36,81	63,19	2.404	4.126
Solidity	26,31	73,69	321	1.220

4.2 Tecnologie più amate e temute secondo SO

In questa sezione viene fatto un confronto dei risultati precedenti (sezione 4.1) con i risultati del sondaggio "Loved vs Dreaded" di Stack Overflow per il 2022.[19]
Siccome i risultati della sezione 4.1 si basano sui post pubblicati dagli utenti mentre i risultati del sondaggio "Loved vs Dreaded" si basano su interviste a sviluppatori realizzate da SO stesso, si possono notare alcune differenze. Alcune tecnologie sono più apprezzate dagli utenti di SO e temute dagli sviluppatori, come ad esempio MATLAB, e viceversa per Kotlin. Come ci sono altre tecnologie apprezzate e temute allo stesso modo, per esempio Python.

Di seguito la percentuale delle tecnologie maggiormente amate e temute più il numero di istanze corrispondenti per ogni tecnologia tra quelle del sondaggio di SO e presenti nella sezione precedente. La tabella è ordinata dalla più alla meno apprezzata.

Tecnologia	**Loved (%)**	**Dreaded (%)**	**Tot. Loved**	**Tot. Dreaded**
Rust	86,73	13,27	5.746	879
TypeScript	73,46	26,54	18.183	6.569
Python	67,34	32,66	22.999	11.156
Go	64,58	35,42	5.116	2.806
SQL	64,25	35,75	22.568	12.559
C#	63,39	36,61	12.603	7.280
Kotlin	63,29	36,71	4.118	2.389
Swift	62,88	37,12	2.194	1.295
Dart	62,15	37,84	2.889	1.759
HTML/CSS	62,09	37,91	24.304	14.838
Solidity	62,08	37,92	640	391
JavaScript	61,46	38,54	28.544	17.899
Bash/Shell	57,89	42,11	11.957	8.699
Scala	50,30	49,70	924	913
Ruby	49,99	50,01	2.149	2.150
C++	48,39	51,61	7.754	8.270

Java	45,75	54,25	10.818	12.827
PowerShell	43,77	56,23	3.753	4.822
PHP	41,61	58,9	6.170	8.657
R	41,60	58,40	1.376	1.932
C	39,68	60,32	5.433	8.259
Assembly	35,91	64,09	1.396	2.491
VBA	21,44	78,56	683	2.502
MATLAB	19,16	80,84	558	2.355

Successivamente le predizioni del modello sono state messe a confronto con il sondaggio di SO per osservare la correlazione. Per ogni linguaggio si confronta la percentuale di emozioni positive secondo il modello con quella del sondaggio di SO. I linguaggi sono confrontati secondo l'ordinamento decrescente per emozioni positive ottenuto dalle predizioni del modello. Per esempio, il primo punto a sinistra rappresenta MATLAB.

Di seguito il grafico che mostra la correlazione.

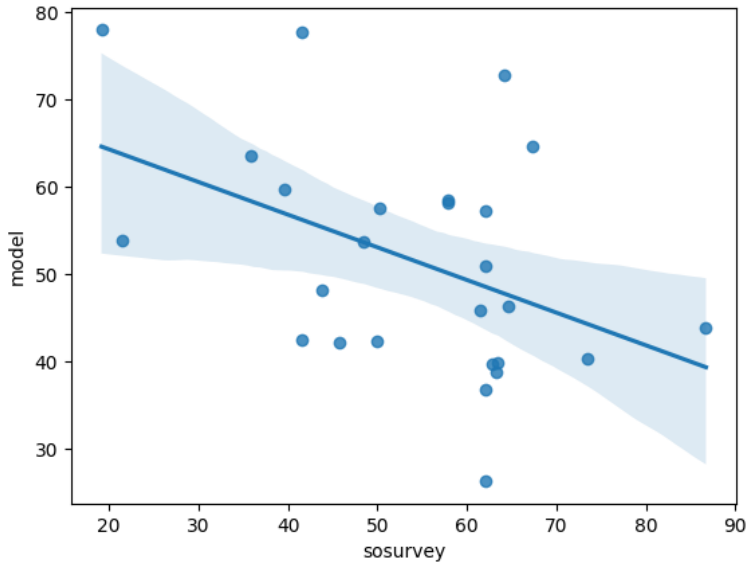

Figure 11: Correlazione tra le predizioni del modello ed il sondaggio di SO

Il coefficiente di correlazione è -0,44 che indica una bassa correlazione. Tale coefficiente varia nell'intervallo [-1;1], ovvero da perfettamente non correlato a perfettamente correlato. Mentre si ha un valore p uguale a 0,02. Questo valore indica la significatività statistica della relazione lineare. Generalmente se inferiore a 0,05 indica che la relazione lineare è statisticamente significativa e non dovuta al caso.

4.3 Emozioni predette dal modello verso le tecnologie più popolari

Dalle predizioni del modello si ottiene che l'emozione positiva principale è *love*, questo probabilmente perché i linguaggi di programmazione e le tecnologie associate permettono di fare cose incredibili al giorno d'oggi, come automatizzare processi ripetitivi tramite programmi. La sensazione che si prova quando si realizza un sistema che svolge compiti in automatico al proprio posto è sicuramente molto positiva e soddisfacente a tal punto di esprimere emozioni positive verso le tecnologie. Mentre l'emozione negativa in assoluto maggiormente espressa è *sad*, che rappresenta la classe sadness, questo per la natura stessa di SO, tipicamente gli utenti pubblicano post perché riscontrano problemi con i linguaggi di programmazione o le tecnologie associate. Quindi, l'emozione negativa può derivare dalla frustrazione di non essere riusciti a risolvere il problema in modo autonomo arrivando a doverlo pubblicare su SO.
L'andamento delle sei emozioni base (*love*, *joy*, *surprise*, *anger*, *fear* e *sadness*) nel corso dell'anno è mediamente costante eccetto in alcuni mesi dove si verifica un'elevata crescita o decrescita. Ad esempio, nel mese estivo luglio si ha una decrescita significativa del numero di istanze dovuta al periodo tipicamente scelto dalla maggior parte degli sviluppatori per le ferie, come nel periodo di dicembre e gennaio, alla fine di questi periodi si ha sempre una crescita delle istanze che tornano mediamente costanti.
Un altro fenomeno che si osserva è l'elevato numero di istanze della classe *neutral*, come accade nella fase di distant supervision, questo è dovuto al fatto che SO è un "social tecnico" dove vengono tipicamente pubblicate domande su problemi riscontrati con i linguaggi di programmazione e le tecnologie associate, di conseguenza non vengono espresse molte emozioni.

Di seguito le emozioni espresse verso le tecnologie più popolari predette dal modello. Gli schemi sono riportati in ordine per numero di istanze totali, maggiori di 5.000, tra emozioni positive e negative.

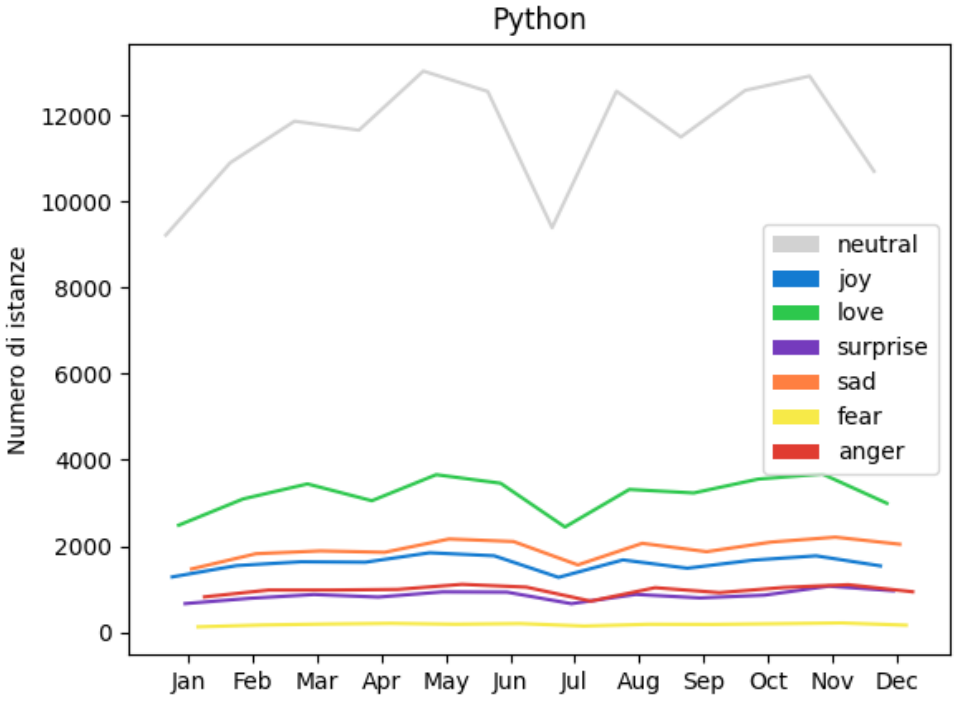

Figure 12: Emozioni predette dal modello verso Python.

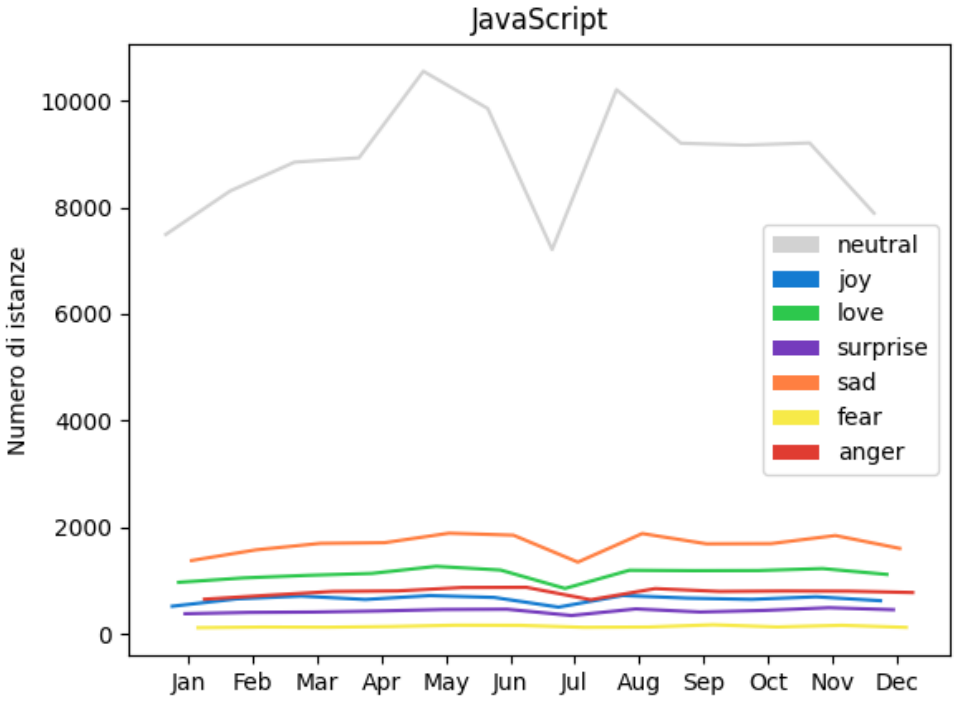

Figure 13: Emozioni predette dal modello verso JavaScript.

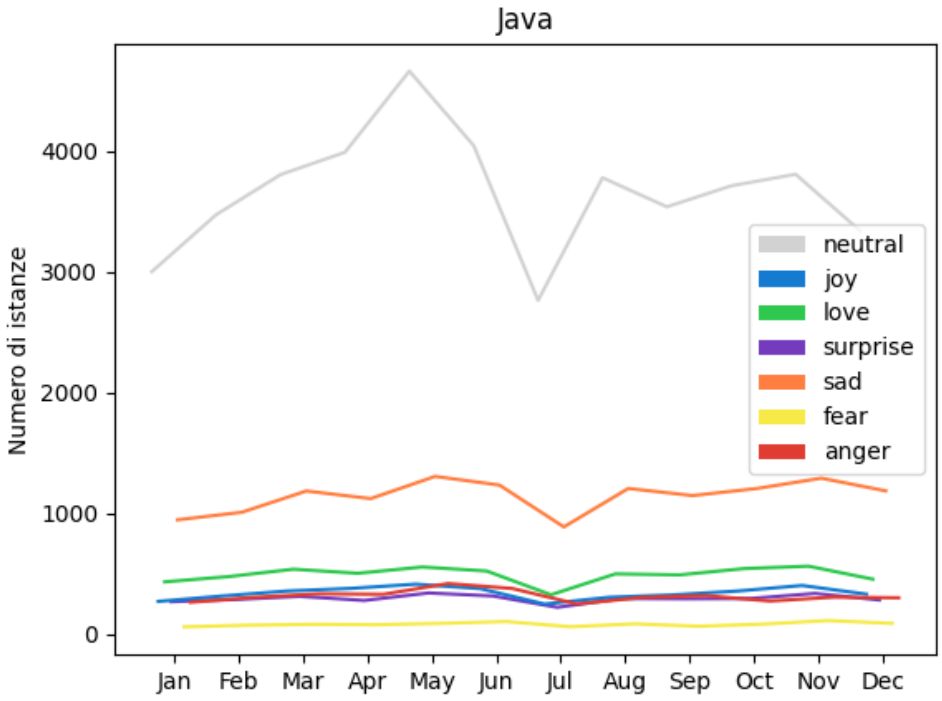

Figure 14: Emozioni predette dal modello verso Java.

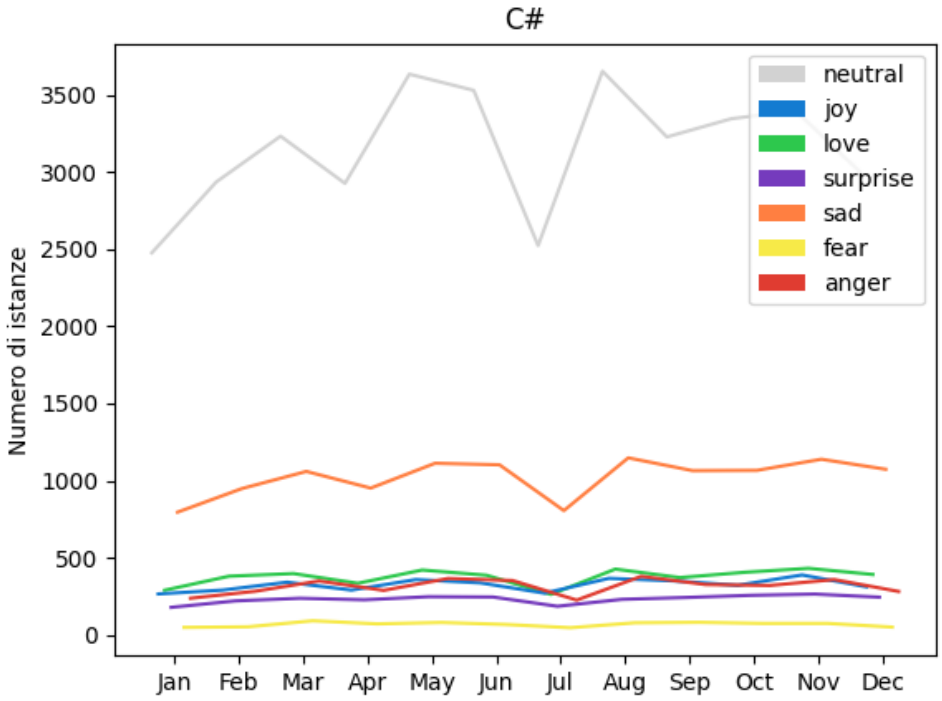

Figure 15: Emozioni predette dal modello verso C#.

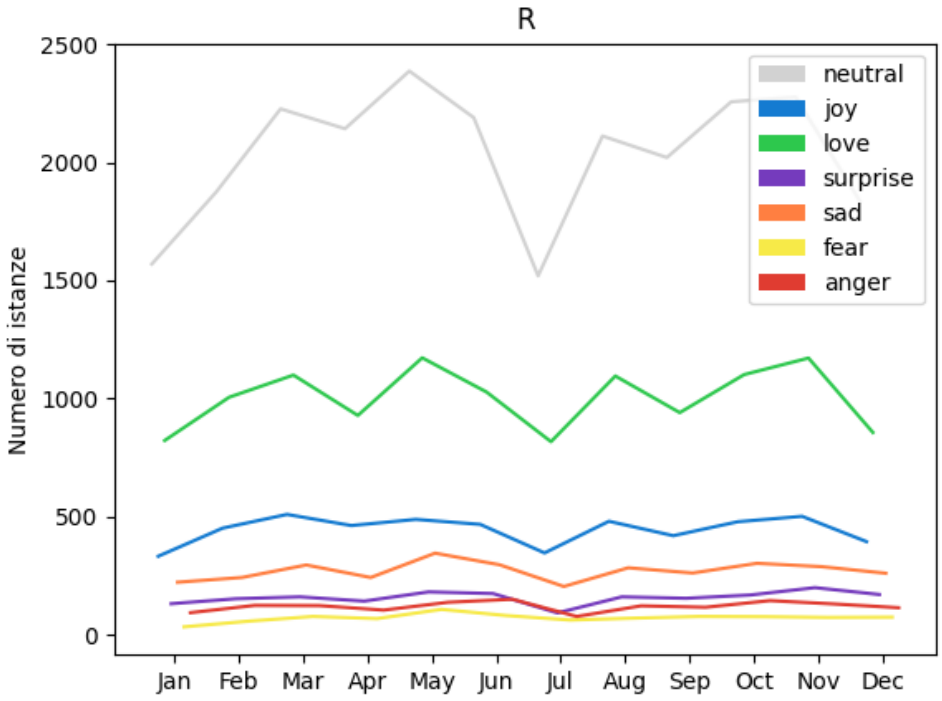

Figure 16: Emozioni predette dal modello verso R.

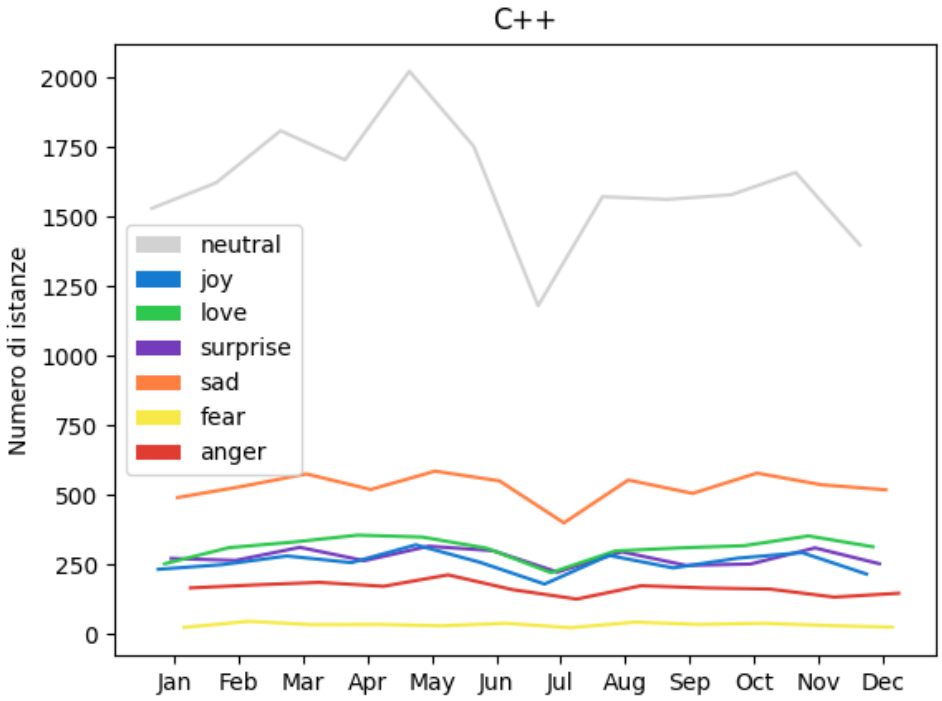

Figure 17: Emozioni predette dal modello verso C++.

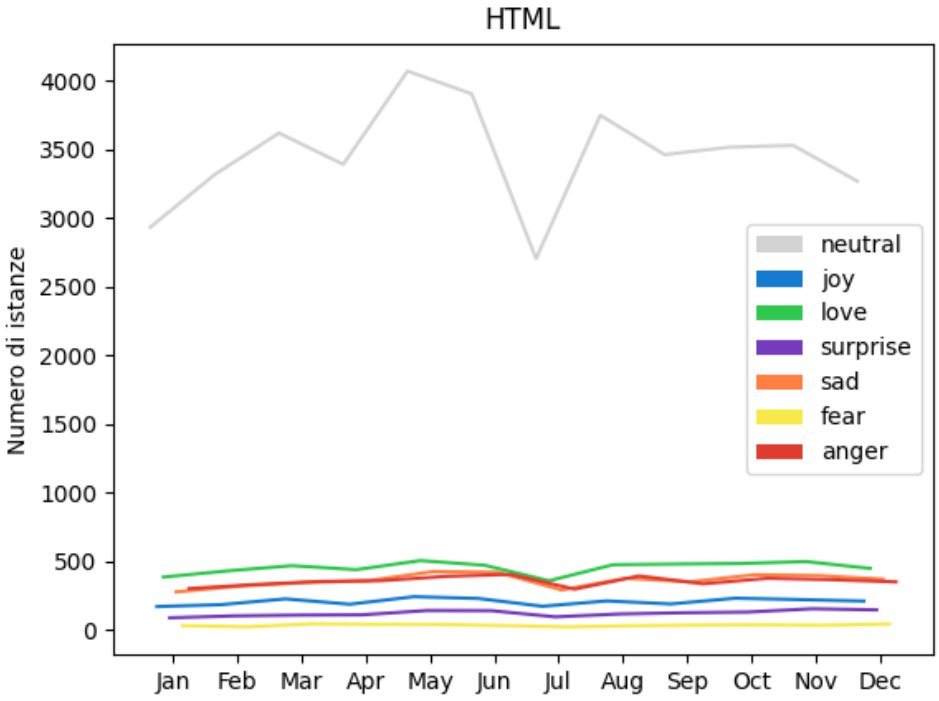

Figure 18: Emozioni predette dal modello verso HTML.

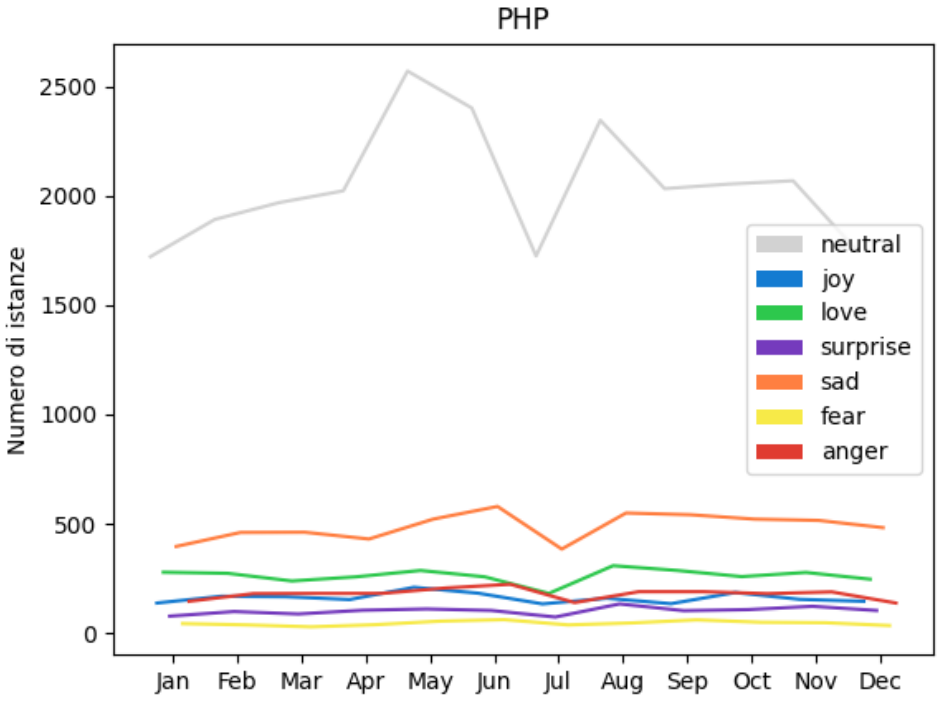

Figure 19: Emozioni predette dal modello verso PHP.

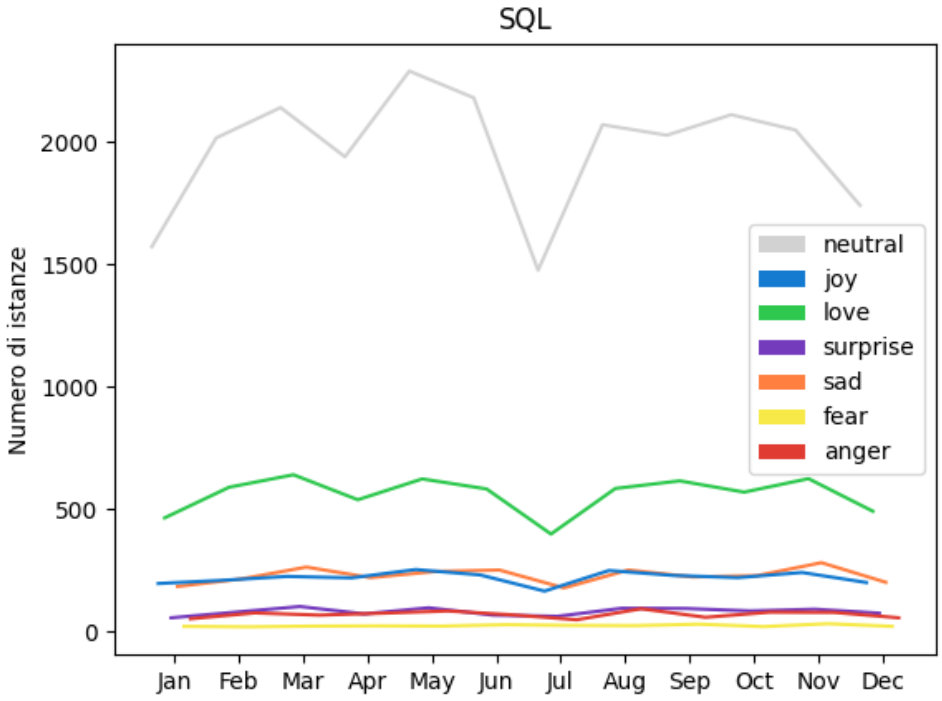

Figure 20: Emozioni predette dal modello verso SQL.

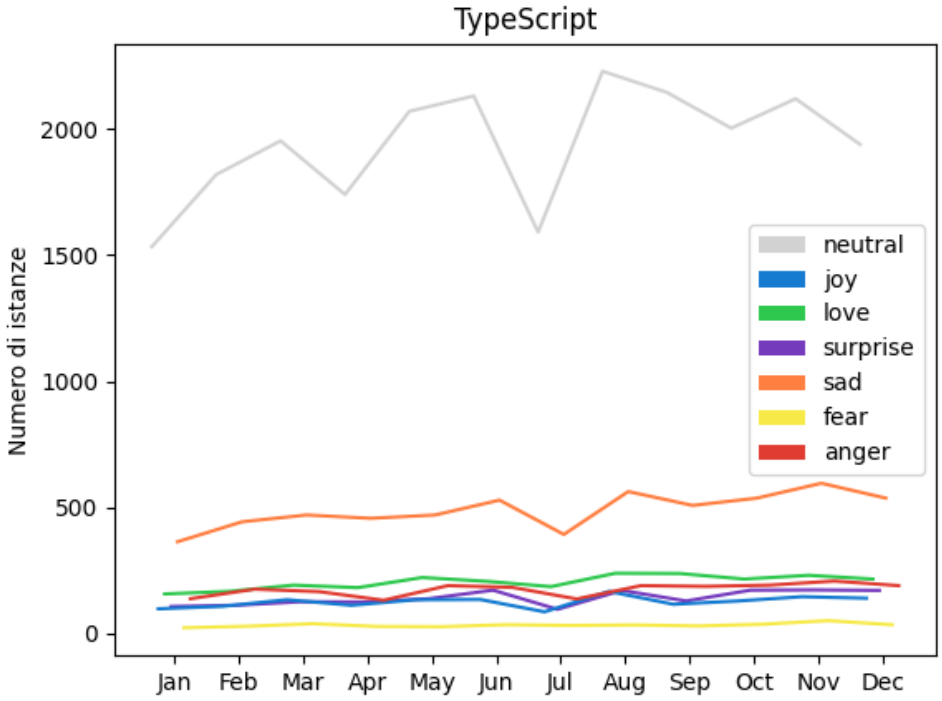

Figure 21: Emozioni predette dal modello verso TypeScript.

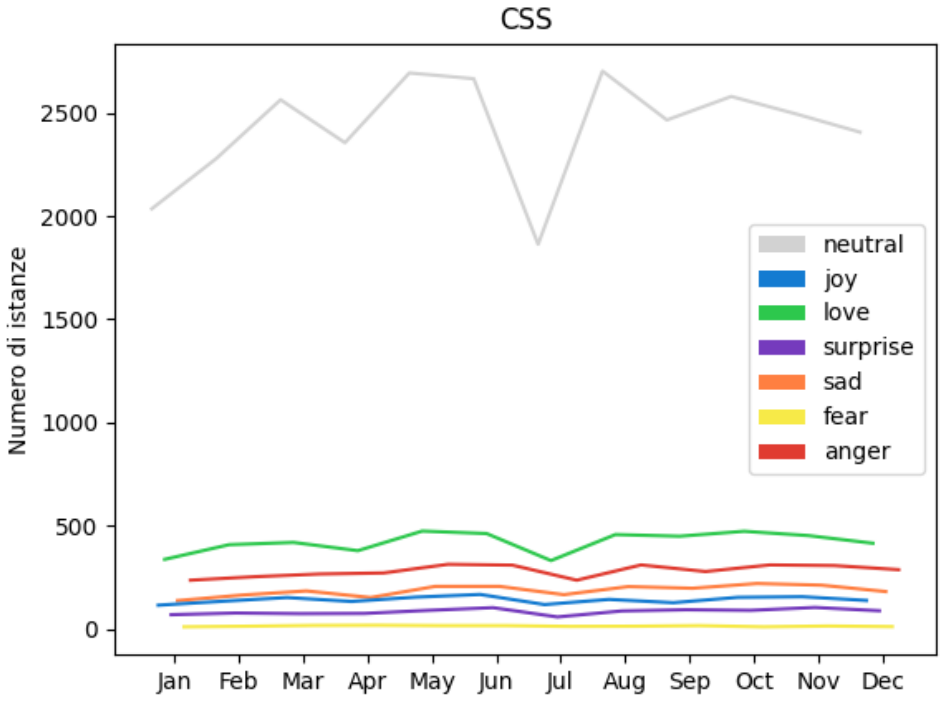

Figure 22: Emozioni predette dal modello verso CSS.

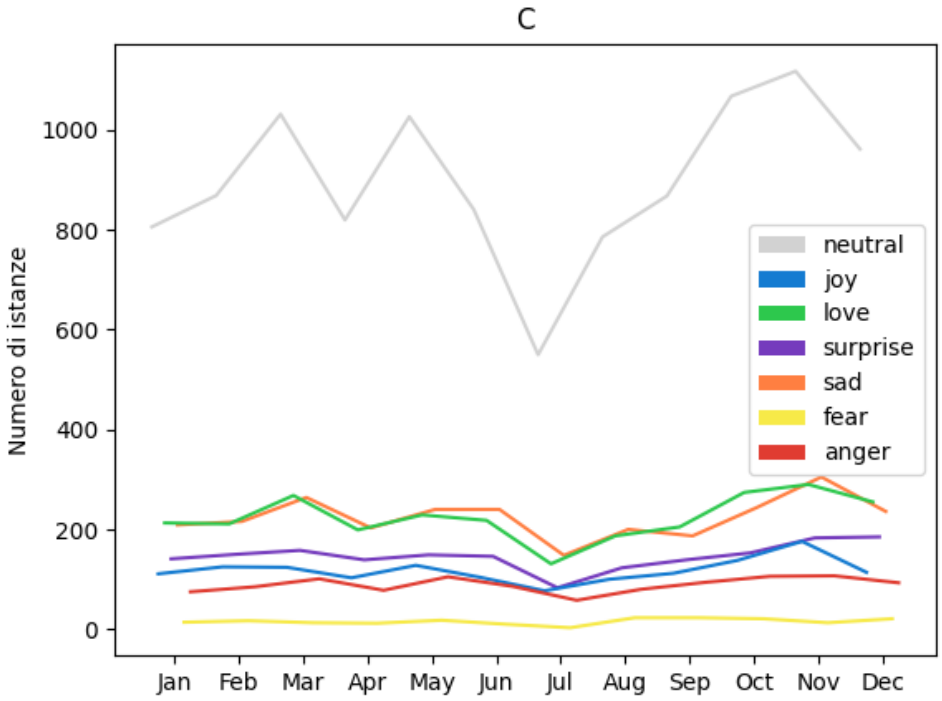

Figure 23: Emozioni predette dal modello verso C.

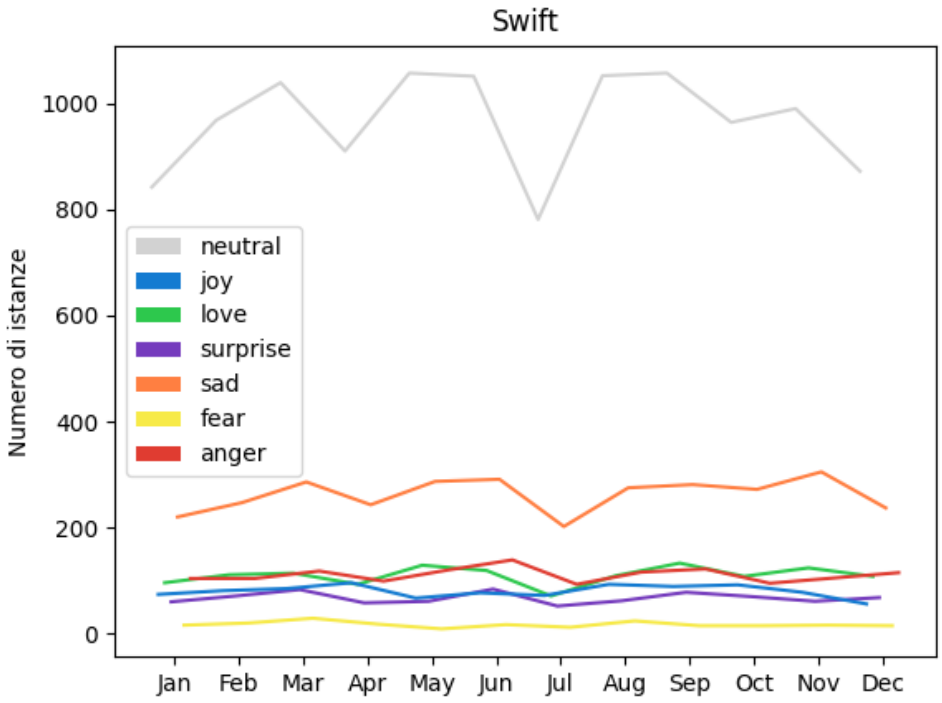

Figure 24: Emozioni predette dal modello verso Swift.

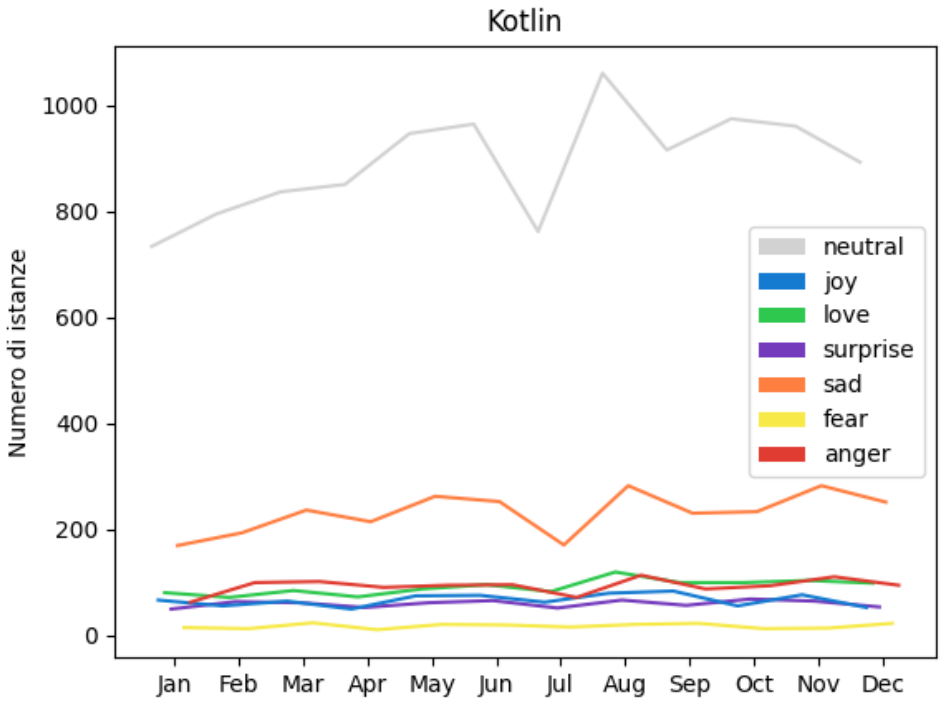

Figure 25: Emozioni predette dal modello verso Kotlin.

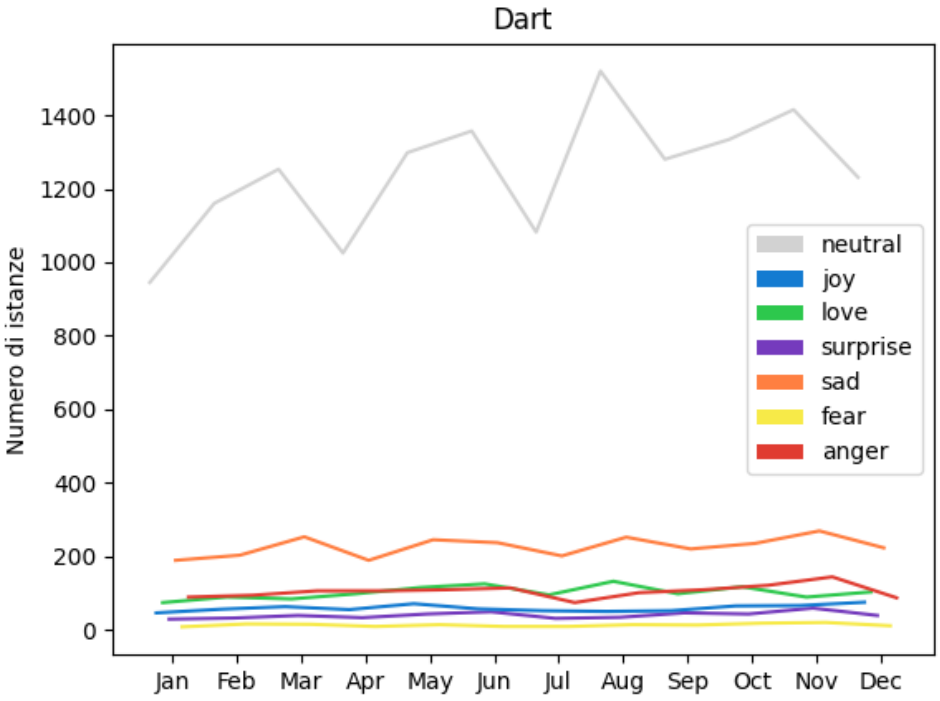

Figure 26: Emozioni predette dal modello verso Dart.

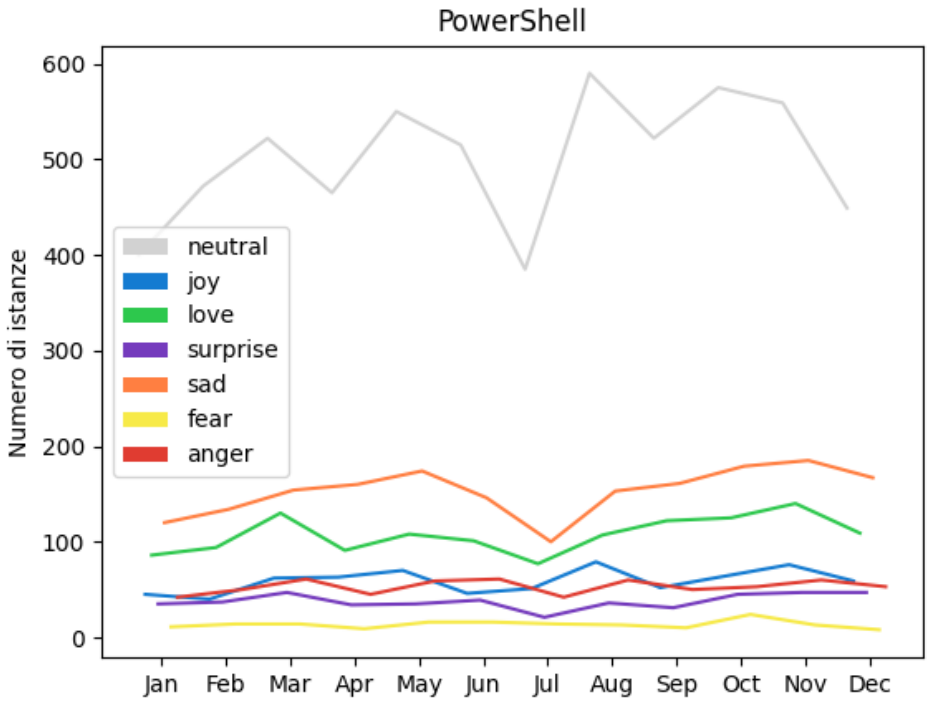

Figure 27: Emozioni predette dal modello verso PowerShell.

5. Conclusione

Normalmente per l'addestramento di modelli tramite apprendimento supervisionato il dataset viene creato a mano etichettando con pazienza tutti gli esempi. Invece in questa tesi è stato utilizzato un sistema di etichettatura automatico secondo il processo di distant supervision, ottenendo nella misura delle prestazioni un'accuratezza media del 66% nello stabilire a quale classe appartiene, tra le sette, il post analizzato. Mentre si ha un'accuratezza del 77% nello stabilire la polarità del post analizzato. Questo è un buon risultato. Per ottenere tale risultato è stato realizzato un modello predittivo basato su BERT addestrato sui contenuti di Stack Overflow, il modello è in grado di classificare correttamente la maggior parte delle emozioni espresse verso i linguaggi di programmazione e le tecnologie ad essi associate come si evince dalle prestazioni. Di conseguenza l'obiettivo primario di questa tesi è stato raggiunto, come l'obiettivo secondario che è stato raggiunto con l'analisi delle predizioni del modello effettuata nel capitolo 4.

Tuttavia, il modello realizzato ha delle limitazioni dovute alla distant supervision, in particolare se nella fase di distant supervision non vengono identificate emozioni o ne vengono identificate in una quantità limitata per alcune classi questo porta ad avere un dataset rumoroso, non bilanciato o di piccole dimensioni. In quanto alla maggior parte delle istanze la distant supervision associa la classe neutrale. Un dataset di piccole dimensioni può non essere sufficiente per un addestramento ottimale, in quanto non ci sono abbastanza esempi per far imparare al meglio la classificazione di tutte le classi di emozioni al modello, di conseguenza non si possono avere risultati eccellenti.

In questa tesi sono state analizzate le domande poste su Stack Overflow mentre le risposte ed i commenti non sono stati analizzati. Questo perché nelle risposte e nei commenti mancano i tag, che sono stati utilizzati per identificare i linguaggi di programmazione e le tecnologie associate. Identificare per ogni risposta e commento la domanda a cui essi fanno riferimento è un compito costoso in termini di tempo e soprattutto capacità computazionale.

Bibliografia

[1] “About Stack Overflow” [Online] https://stackoverflow.co/.

[2] “Stack Overflow” [Online] https://en.wikipedia.org/wiki/Stack_Overflow.

[3] “Programming language” [Online] https://en.wikipedia.org/wiki/Programming_language.

[4] “Intelligenza artificiale” [Online] https://en.wikipedia.org/wiki/Artificial_intelligence.

[5] “Machine learning” [Online] https://en.wikipedia.org/wiki/Machine_learning.

[6] “Apprendimento non supervisionato” [Online] https://it.wikipedia.org/wiki/Apprendimento_non_supervisionato.

[7] “Apprendimento supervisionato” [Online] https://it.wikipedia.org/wiki/Apprendimento_supervisionato.

[8] “Distant supervision” [Online] http://deepdive.stanford.edu/distant_supervision.

[9] “Natural Language Processing” [Online] https://en.wikipedia.org/wiki/Natural_language_processing.

[10] “BERT” [Online] https://it.wikipedia.org/wiki/BERT.

[11] “BERT (language model)” [Online] https://en.wikipedia.org/wiki/BERT_(language_model).

[12] “Transformer ML model” [Online] https://en.wikipedia.org/wiki/Transformer_(machine_learning_model).

[13] “Overfitting” [Online] https://it.wikipedia.org/wiki/Overfitting.

[14] “Sentiment analysis” [Online] https://it.wikipedia.org/wiki/Analisi_del_sentiment.

[15] “Classificazione di emozioni” [Online] https://en.wikipedia.org/wiki/Emotion_classification.

[16] “Archive Stack Exchange” [Online] https://archive.org/download/stackexchange.

[17] “Databases Stack Exchange” [Online] https://data.stackexchange.com/.

[18] “Stack Overflow Developer Survey 2022” [Online] https://survey.stackoverflow.co/2022/#technology-most-popular-technologies.

[19] “Stack Overflow survey Loved vs Dreaded” [Online] https://survey.stackoverflow.co/2022/#most-loved-dreaded-and-wanted-language-love-dread.