Progetto Deep Learning: Sarcasm Detection with SLM & Multitask Learning

Domenico De Marchis - 578132, Francesco Fontanesi - 578366

January 2026

1 Introduzione

Il progetto si concentra sullo studio del task di **Sarcasm Detection**. L'obiettivo è confrontare le performance di Small Language Models (SLM) specializzati tramite fine-tuning rispetto a Large Language Models (LLM) utilizzati in modalità zero-shot. I modelli presi in esame sono:

- **Encoder-only**: ModernBERT, evoluzione moderna del modello BERT disponibile su HuggingFace, ideale per compiti di classificazione testuale sfruttando la bidirezionalità dell'attention
- **Decoder-Only (SLM)**: Phi-3, modello open-source orientato alla generazione
- **Large Language Models**: Llama-3.3-70b-versatile e Kimi-K2 disponibili tramite l'API di Groq, utilizzati come benchmark

La difficoltà principale sta nella natura stessa del sarcasmo: molto spesso servono informazioni contestuali per comprenderlo, non sempre disponibili e generalmente il concetto stesso è difficile anche per esseri umani molte volte. L'obiettivo dell'esperimento è verificare se un encoder allenato con strategie ad hoc per il problema (apprendimento multitask e tecniche per gestire lo sbilanciamento delle classi, vedi dopo) possa superare modelli molto più grandi in zero - shot e che, come mostrano i dati preliminari, tendono a sovrastimare la presenza di sarcasmo; i LLM utilizzati come benchmark forniscono comunque una baseline da battere, anche perché il task è sì difficile ma la conoscenza generalista ed il prompting li rende viabili, in realtà, anche per task molto più complessi di questo.

1.1 Dataset - iSarcasmEval

Il task di riferimento è una semplice classificazione binaria sulla presenza, o meno, di sarcasmo nel testo. Se si guarda a task concettualmente simili, si potrebbe obiettare che una classificazione binaria sia troppo semplice: in rete

si trovano anche situazioni, soprattutto sulla classificazione di emozioni, dove il task comprende un numero molto alto di classi, 28 addirittura per alcuni e questo, spesso e volentieri, confonde i modelli lasciando poco segnale. Una classificazione binaria va bene perché il problema sul sarcasmo ed emozioni sta proprio nei dati: non esistono etichette 'reali' e molte dipendono dall'intuizione di chi etichetta, il sarcasmo stesso, molte volte, dipende dall'autore e non ci sono linee chiare. Il dataset scelto è **iSarcasmEval**:

- **Struttura**: il dataset presenta uno sbilanciamento significativo, con un rapporto di circa **1:3** tra classe sarcastica e non sarcastica; i record sono principalmente tweets
- **Vocabulary Mismatch**: abbiamo deciso deliberatamente di non integrare dati esterni provenienti da fonti come Reddit. Sebbene aumenterebbe la quantità di dati, introdurrebbero un forte *vocabulary mismatch* (slang, stile di scrittura differente) che rischierebbe di aumentare il rumore
- **Rephrasing**: una peculiarità fondamentale di questo dataset è la presenza, per molti esempi sarcastici, di una versione parafrasata in linguaggio non sarcastico. Questa caratteristica è la base che abilita il nostro approccio Multitask

Gli stessi autori dedicano molto tempo alla fase preliminare di data augmentation prima di addestrare i modelli proposti, arricchendo il dataset con commenti Reddit, battute da serie TV... Come già detto, iSarcasmEval è composto prevalentemente da tweet e, a nostro avviso, questa scelta potrebbe non essere ottimale poiché introduce un vocabolario e uno stile linguistico sensibilmente diversi rispetto al dominio di partenza, o meglio, non puoi aspettarti semplicemente di 'unire' i dataset ma dovresti riuscire in qualche modo a quantificare gli esempi stilisticamente simili e ci vuole tempo. Per questo motivo, abbiamo preferito limitare l'intervento sui dati e concentrarci invece sulla formulazione del problema e sull'impostazione metodologica, con l'obiettivo di massimizzare le prestazioni.

1.2 Training

Per massimizzare le prestazioni, abbiamo adottato due strategie chiave per mitigare lo sbilanciamento delle classi e la scarsità di dati:

- **Weighted Loss**: Dato il rapporto circa 1:3 tra esempi positivi (sarcastici) e negativi, durante il training viene assegnato un peso maggiore alla loss calcolata sui record positivi
- **Approccio Multitask**: Per arricchire il segnale di addestramento, non ci limitiamo alla classificazione binaria sulla singola frase. Costruiamo un *Task B* ausiliario utilizzando le coppie $(text, rephrase)$ fornite dal dataset. Il modello deve identificare quale delle due frasi è sarcastica; questo permette di propagare il gradiente non solo rispetto al task originario, ma

anche basandosi sulla differenza semantica tra la frase sarcastica e la sua riformulazione letterale.

Non è l'unico task ausiliario che abbiamo provato: sempre sfruttando la coppia $(text, rephrase)$, è possibile utilizzare un approccio di **Contrastive Loss** (o **Margin Loss** nel nostro caso) che, calcolando uno score di sarcasmo per entrambe le frasi, miri a massimizzare la distanza tra le loro rappresentazioni. L'obiettivo è imporre che il punteggio assegnato al testo originale sia sistematicamente superiore a quello della parafrasi, 'spingendo' il modello a distinguere le sfumature semantiche. La loss complessiva è definita, quindi, come una combinazione pesata di due termini:

$$\mathcal{L} = \lambda_A \mathcal{L}_A + \lambda_B \mathcal{L}_B \tag{1}$$

dove $\mathcal{L}_A$ è una **BCEWithLogitsLoss** sul task A (classificazione binaria), eventualmente configurata con un peso per le classi positive (`pos_weight`) per gestire lo sbilanciamento. L'allenamento multitask avviene mischiando in ogni batch esempi dei task A e B (approssimativamente metà e metà) tramite un *batch sampler*. Nel training step la loss viene calcolata solo per i task effettivamente presenti nel batch corrente e si esegue un unico passaggio di backward e di ottimizzazione sui parametri condivisi. In questo senso l'impostazione è un hard parameter sharing: backbone (e head) sono condivisi, e gli score sono logit reali prodotti dalla stessa testa lineare, riutilizzati come logit per la BCE (task A) e come punteggi da confrontare nella MarginRankingLoss (task B). Per il task B si utilizza una **MarginRankingLoss** con margine m. Dati gli score s_{sarc} e s_{reph}, si impone la condizione $s_{\text{sarc}} \geq s_{\text{reph}} + m$ e la loss risultante è:

$$\mathcal{L}_B = \max(0,\, m - (s_{\text{sarc}} - s_{\text{reph}})) \tag{2}$$

che penalizza il modello solo quando la differenza tra i due score non supera il margine m. Questo parametro m è nella scala dei logit.

2 Risultati Preliminari: LLM Zero-Shot

Per stabilire una baseline, abbiamo testato due modelli di grandi dimensioni in modalità zero-shot. I risultati mostrano un fenomeno interessante: i modelli grandi hanno una *Recall* altissima (trovano quasi tutto il sarcasmo) ma una *Precision* molto bassa. In pratica, classificano come sarcastica una vastissima quantità di esempi, generando moltissimi falsi positivi.

Table 1: Performance Zero-Shot su iSarcasmEval (Classe 'Sarcastic')

Modello	**F1-Score**	**Precision**	**Recall**
Llama-3.3-70b	0.3472	0.2110	**0.9800**
Kimi-K2	0.4008	0.2526	0.9700

Dalle matrici di confusione si osserva che entrambi i modelli tendono a sovrastimare la classe *Sarcastic*: **Llama-3.3-70b** genera 733 falsi positivi su 1200 esempi non sarcastici, mentre **Kimi-K2** riduce solo parzialmente il fenomeno (574 FP). Questo comportamento è coerente con i risultati zero-shot in Tabella 1, dove il **recall** è quasi massimo (0.98 e 0.97) ma la **precision** rimane bassa (0.211 e 0.253), producendo F1 contenuti. Con **ModernBERT** puntiamo quindi a contenere i falsi positivi, anche a costo di una lieve riduzione del recall, per ottenere una precisione (e quindi un F1) più utile.

3 ModernBERT

ModernBERT si colloca nella linea encoder-only di BERT, ma introduce scelte più recenti a livello di architettura ed addestramento, con l'obiettivo di rendere il backbone più efficiente, stabile e, soprattutto, più efficace in fase di fine-tuning. Nel nostro caso viene fine-tuned su **iSarcasmEval** per mitigare il bias osservato nei modelli zero-shot, che tendono a massimizzare il *recall* della classe *Sarcastic* a costo di una *precision* molto bassa (e quindi molti falsi positivi). I risultati mostrano infatti un cambio di comportamento: rispetto a **Llama-3.3-70b** e **Kimi-K2** (recall $\approx$ 0.98/0.97 ma precision 0.211/0.253), ModernBERT produce predizioni più selettive. Con `modernbert_o2` (soglia 0.50) la precision cresce a 0.314, con un recall più contenuto (0.38); aumentando la soglia a 0.75 (`modernbert_o2_second`) si modifica il trade-off, incrementando la copertura (recall 0.555) ma con una precision ancora limitata (0.262). ModernBERT, quindi, riduce l'"aggressività" tipica dello zero-shot e, comunque, raggiunge performance comparabili a quelle di modelli che hanno un numero spropositatamente maggiore di parametri. D'altronde il training di **ModernBERT** è stato eseguito con tecniche **PEFT** come **LoRA** ed il numero di parametri che abbiamo cambiato è circa lo 0.7% del totale.

Table 2: Confronto tra Zero-Shot e ModernBERT su iSarcasmEval (classe *Sarcastic*)

Modello	**Parametri**	**F1**	**Precision**	**Recall**
Llama-3.3-70b (zero-shot)	70B	0.3472	0.2110	0.9800
Kimi-K2 (zero-shot)	1T	0.4008	0.2526	0.9700
ModernBERT `o2` (th=0.50)	149M	0.3439	0.3140	0.3800
ModernBERT `o2_second` (th=0.75)	149M	0.3563	0.2624	0.5550

3.1 Threshold Tuning

Nel nostro caso il validation set è piccolo e rumoroso: lo manteniamo per monitorare l'overfitting durante il training, ma non è affidabile per tarare in modo robusto una soglia decisionale. In inferenza adottiamo quindi la scelta standard

$t = 0.5$: non è ottimale, ma è una decisione vincolata dalle condizioni sperimentali. Durante il training, un modello tipo BERT impara a produrre logit più alti per la classe positiva e più bassi per la negativa, ma la conversione "logit $\rightarrow$ classe" richiede comunque una threshold. In dataset bilanciati, i contributi (gradienti) di positivi e negativi tendono a compensarsi e la distribuzione dei logit risulta più facilmente separabile con $t = 0.5$. Qui però le classi sono sbilanciate, quindi per ottenere un decision boundary più coerente con la soglia usata in inferenza abbiamo aumentato il peso dei positivi in loss tramite `pos_weight`, proporzionale allo sbilanciamento e questo è necessario a 'pareggiare' il contributo dei gradienti.
Con BCE + `pos_weight` $= W$ stiamo dicendo che gli errori sui positivi valgono W volte di più, introducendo un bias voluto nei logit. Per recuperare la probabilità "non pesata" p_{real} dalla probabilità del modello pesato p_{weighted} vale:

$$p_{\text{real}} = \frac{p_{\text{weighted}}}{p_{\text{weighted}} + (1 - p_{\text{weighted}})\, W} \tag{3}$$

Imponendo $p_{\text{real}} = 0.5$ si ottiene che il punto neutrale si sposta a:

$$p_{\text{weighted}} = \frac{W}{1 + W} \tag{4}$$

Con $W = 3.0$:

$$\frac{3}{1+3} = \frac{3}{4} = 0.75$$

In altre parole, `pos_weight`$= 3.0$ giustifica teoricamente una soglia "neutrale" attorno a 0.75. Tale $W = 3$ è indicato come esempio poiché, tra le configurazioni, è quello che ha prodotto i migliori risultati.

3.2 ModernBERT Multitasking

I risultati mostrano chiaramente che il primo task ausiliario (predire quale tra *sentence* e *rephrase* fosse "sarcastico") è poco informativo: `modernbert_o1` produce un comportamento estremamente "aggressivo" (recall 0.96) ma con precision molto bassa (0.141), segnale di moltissimi falsi positivi. Al contrario, il task contrastivo con **Margin Loss** fornisce un segnale di apprendimento più utile: l'ablazione conferma che rimuoverlo peggiora sensibilmente le prestazioni. In particolare, mantenendo **stessa configurazione** di `modernbert_o2_second` (il migliore, stessa soglia $t = 0.75$, stesso training setup), il modello `only_task_a` scende a F1=0.248, mentre la versione multitask con Margin Loss arriva a F1=0.36. Questi sono risultati 'finali' e che utilizziamo per spiegare, tuttavia l'andamento era già visibile su poche epoche in fase di costruzione e difatti ci siamo concentrati prevalentemente sul secondo task.

Table 3: Confronto tra task ausiliari e ablazione (classe *Sarcastic*)

Setup	Soglia	F1	Precision	Recall
`modernbert_o1`	0.48	0.245	0.141	0.960
`modernbert_o2_only_task_a`	0.75	0.248	0.174	0.430
`modernbert_o2_second`	0.75	**0.356**	0.262	0.555

3.3 Tentativi ed Ablazione

Oltre allo studio sull'utilità del task ausiliario, abbiamo eseguito una serie di ablazioni successive mirate a quantificare l'efficienza del (`pos_weight`).

Ablazione completa: rimozione di pesi e task B. Nel setup `o2_ablation_all` abbiamo rimosso sia il task ausiliario (Margin Loss) sia la pesatura dei positivi (`pos_weight`), tornando ad una BCE standard e decisione con soglia $t = 0.5$. Il modello collassa verso la classe maggioritaria: l'accuracy rimane alta (0.84) ma la copertura della classe sarcastica diventa trascurabile (recall 0.05, F1 0.083). In pratica, su 200 esempi sarcastici il modello ne recupera circa 10, predicendo pochissimi positivi complessivi (circa 42), quindi con un recall troppo basso per essere utile; appunto: lo sbilanciamento tira il modello a prevedere score bassissimi relativi alla classe negativa, difatto la maggioranza e non prevedendo mai il sarcasmo.

Ablazione dei soli pesi di classe. Nel setup `o2_ablation` abbiamo rimosso `pos_weight` mantenendo il resto invariato. Si osserva un comportamento più conservativo: la precision sale fino a 0.40, ma il recall rimane basso (0.11) e l'F1 si ferma a 0.173. Questo risultato è coerente con l'interpretazione dei pesi di classe come meccanismo per contrastare la tendenza del modello ad "assorbire" la classe maggioritaria, migliorando la copertura dei positivi.

Sensibilità al peso del task ausiliario. Abbiamo inoltre esplorato la sensibilità ai coefficienti della loss multitask, aumentando il peso del task B (Margin Loss) fino a triplicarlo rispetto alla configurazione ottima. Nel setup `o2_third` (con soglia $t = 0.75$, coerente con `pos_weight` ≈ 3) si osserva un *shift* del comportamento rispetto al modello migliore: il recall cresce (0.36), ma la precision si riduce (0.23) e l'accuracy cala sensibilmente (0.736). Questo indica che il task B è informativo e può effettivamente aumentare la sensibilità al sarcasmo, ma se sovrapesato introduce un bias verso predizioni più "aggressive", con un incremento dei falsi positivi e quindi una perdita di selettività. Il task contrastivo è, quindi, utile, ma richiede un bilanciamento accurato con il main task A: oltre un certo punto, il vantaggio dato dal ragionamento sugli score si annulla e diventa rumore ulteriore per il task principale

Table 4: Ablazioni su pesi di classe e pesatura del task ausiliario

Setup	Soglia	Accuracy	F1	Precision	Recall
`o2_ablation_all`	0.50	0.841	0.083	0.238	0.050
`o2_ablation`	0.75	0.849	0.173	0.400	0.110
`o2_third`	0.75	0.736	0.281	0.230	0.360

Altri tentativi (learning rate, margine, LoRA). Abbiamo condotto ulteriori esperimenti variando learning rate, margine m della MarginRankingLoss e configurazioni **PEFT/LoRA**. In tutti i casi, il training mostra un comportamento coerente con la dimensione ridotta del dataset: le prestazioni saturano rapidamente e difficilmente migliorano oltre $\sim$5 epoche, con segnali di overfitting abbastanza chiari guardando l'andamento della loss sul validation set; difatti, nonostante la rumorosità di questo, risulta comunque importante utilizzarlo per monitorare il training stesso. Le variazioni di iperparametri non hanno prodotto un guadagno stabile rispetto alla configurazione migliore e ci fermiamo ad una $F_1 = 0.36$, che è un buon risultato considerando che il BERT proposto dagli autori si ferma a $F_1 = 0.34$ ma su molti più dati.
Come ulteriore controllo, abbiamo valutato anche un esperimento di **Full Fine-Tuning** (`o2_fft`), in cui l'intero backbone viene aggiornato (senza LoRA) mantenendo la stessa soglia $t = 0.75$. Il modello mostra un comportamento più instabile: tende a non reggere oltre $\sim$2 epoche senza overfitting, segnale coerente con l'elevata capacità del backbone rispetto alla quantità di dati disponibili. Sul test, FFT produce un aumento del recall (0.495) ma a fronte di una precision limitata (0.236) e un calo dell'accuracy (0.699), evidenziando un trade-off simile a quello osservato quando il modello diventa più "aggressivo" nelle predizioni. Nel complesso quindi, questo rafforza l'idea che **LoRA sia un'alternativa molto viabile** su dataset piccoli: consente di ottenere buoni risultati con relativamente pochi parametri e, proprio questo concetto, permette di allenare il modello su più epoche nel caso di scarsità.
Le ablazioni e i tentativi indicano quindi che:

- la sola BCE non pesata tende a favorire la classe maggioritaria, penalizzando il recall sulla classe sarcastica
- il task B con Margin Loss fornisce un segnale utile ma va bilanciato, un peso eccessivo induce uno shift verso predizioni più "aggressive" e incrementa i falsi positivi
- il Full Fine-Tuning, pur aumentando la recall, risulta più soggetto ad overfitting e meno stabile rispetto a LoRA con pochi dati come nel nostro caso

4 Phi-3

Phi-3-mini-4k-instruct è un modello **decoder-only** generativo. A differenza delle architetture basate su encoder (come ModernBERT), che prevedono l'uso di una *classification head* dedicata, Phi-3 sfrutta l'autoregressività per predire il token successivo. Ciò comporta una sostanziale differenza nella costruzione del dataset e nella fase di inferenza, pur mantenendo l'obiettivo finale di classificazione.
Nel nostro setup sperimentale, la rilevazione del sarcasmo viene affrontata secondo due modalità:

- **Predizione generativa diretta**: il modello viene istruito a generare esplicitamente l'etichetta di classificazione (es. A per sarcastico, B per non sarcastico). Questo approccio è quello nativo per un LLM.
- **Analisi dei Logit** (approccio discriminativo): si valutano le probabilità (logit) assegnate dal modello ai token target (es. ℓ_A vs ℓ_B). La decisione finale viene presa applicando una soglia τ sul differenziale $\Delta = \ell_A - \ell_B$.

  Quando si applica un `pos_weight` ai campioni, come prima bisogna comunque portare in conto che la soglia naturale, in questo caso Δ viene anch'essa shifatata come prima. In questo caso, applicando le stesse relazioni, risulta che il confine naturale passa da $0.0 \rightarrow 0.25$

Il secondo approccio permette di ottimizzare la sensibilità del classificatore tramite *thresholding*. Come mostrato in Tabella 5, l'innalzamento della soglia (*Threshold Upshift*) incrementa notevolmente le performance rispetto alla configurazione di default.

Table 5: Confronto Metriche Sarcasm Detection (Class 1)

Metric	**Phi-3 Generative** *(Default Threshold)*	**Threshold Tuned** *(Upshifted Threshold)*
Precision	0.2416	**0.4755**
Recall	**0.7900**	0.4850
F1-Score	0.3700	**0.4802**

È tuttavia necessario notare che la configurazione *Threshold Tuned* presentata è stata ricavata ottimizzando manualmente la soglia sul test set. Questa procedura introduce un bias di *data leakage* (o *peeking*); pertanto, tali valori non vanno considerati come una baseline operativa, ma come una stima del **potenziale teorico** che il modello potrebbe esprimere tramite un fine-tuning supervisionato che calibri correttamente i logit.

4.1 Strategia Multitasking

Essendo Phi-3 un modello *instruction-tuned*, il processo di fine-tuning differisce dal classico apprendimento supervisionato su coppie `(input, label)`. I dati

vengono strutturati come interazioni **prompt** $\rightarrow$ **completamento**, coerenti con il formato di addestramento del modello. La strategia multitask adottata combina due obiettivi distinti:

- **Task A (Classificazione)**: richiede la generazione di un singolo token ("A" o "B"), difatto main task
- **Task C (Rewriting)**: richiede la riscrittura del testo sarcastico in una forma *letterale e non sarcastica*, preservandone il significato semantico

Il task C, in questo caso, viene concettualmente dal task B costruito per ModernBERT; abbiamo deciso di non utilizzare lo stesso esatto task di prima (B) poiché non sarebbe stato coerente con le nuove condizioni: abbiamo un modello generativo, usiamo questa capacità.

Definizione dei Prompt. Riportiamo i prompt utilizzati per i due task; ricordiamo che il modello è uno SLM, quindi meglio prompt semplici e chiari, più 'standard'.

Task A (Sarcasm Detection)

```
Classify the following text.
Reply ONLY with "A" if it is sarcastic, or "B" if it is NOT sarcastic.
Text: <text>
Answer:
```

Task C (Non-sarcastic Rewrite)

```
Rewrite the text into a literal, non-sarcastic version while preserving
the meaning.
Text: <text>
Rewrite:
```

I target attesi sono:

- **Task A**: il token `"A"` (se sarcastico) o `"B"` (se non sarcastico), seguito da `<eos>`
- **Task C**: la frase riformulata (*rephrase*) fornita dal dataset, seguita da `<eos>`

Loss Masking e Chat Template. Per preservare le capacità instruction - tuned del modello, utilizziamo il **chat template** nativo di Phi-3 (incluso il system message standard). Durante il training, la loss viene calcolata esclusivamente sui token del **target** (la risposta del modello), mascherando i token del prompt tramite l'assegnazione di `IGNORE_INDEX=-100` nel vettore delle label. Questo è importante perché il prompt fornisce l'istruzione, non il target di generazione.

Integrazione del Task C. Il Task C viene attivato esclusivamente per gli esempi positivi (`label=1`) per i quali è disponibile una **rephrase** nel dataset. Come suddetto, l'idea multitask è la stessa del caso precedente: oltre che imparare il concetto di sarcasmo dalla rappresentazione binaria, la loss sulla scrittura dei token permette di separare il sarcasmo da ciò che non lo è in maniera più netta.

Funzione di Loss Combinata. Il training avviene su un batch misto di esempi, gestito come nel caso di ModernBERT. La funzione di loss totale è una somma pesata delle componenti:

$$\mathcal{L} = \lambda_A \mathcal{L}_A + \lambda_C \mathcal{L}_C$$

dove $\mathcal{L}_A$ e $\mathcal{L}_C$ rappresentano la cross-entropy rispettivamente sul token di classificazione e sulla sequenza di riscrittura. Abbiamo sperimentato configurazioni con $\lambda_C = \lambda_A$ e $\lambda_C = 0.5\lambda_A$, vedendoc che, come nel caso precedente di ModernBERT, il task ausiliario è sì utile ma se si mantiene chiara l'importanza maggiore del main task. In realtà, con il fatto che il task C è generativo, abbiamo deciso di adottare una loss normalizzata per evitare che sequenze molto lunghe di rephrase siano avvantaggiate e generino un gradiente sproporzionato rispetto alla media del training set.

$$\mathcal{L} = \frac{\sum_{i,t} w_i \cdot 1_{\text{active}}(i,t) \cdot \text{CE}(\hat{y}_{i,t}, y_{i,t})}{\sum_{i,t} w_i \cdot 1_{\text{active}}(i,t)}$$

4.2 LoRA e SwiGLU

Nel fine-tuning di **Phi-3** utilizziamo ancora **PEFT/LoRA**, ma con una differenza rispetto a **ModernBERT**: oltre alle proiezioni dell'attention, applichiamo LoRA anche alle proiezioni della **MLP** interna ai blocchi Transformer. La motivazione è che, a livello architetturale, differentemente da BERT il modello Phi-3 fa dell'adozione di SwiGLU il grosso delle innovazoni, quindi ci sembrava doveroso ragionare su questa.
Invece di una singola proiezione, l'MLP produce due rami di trasformazioni lineari: uno che genera le feature e uno che funge da *gate*, applicando le due proiezioni separatamente ad entrambe le metà della sequenza iniziale. Solamente una di queste vede l'applicazione di non linearità, che nel caso di SwiGLU è una **Swish**, avendo come output del layer complessivamente il dot product tra la metà della sequenza che vede solo la proiezione lineare e l'altra metà che vede la non - linearità, gate appunto; meccanismo simile alle LSTM se ci si pensa. Il training si concentrerà, quindi, sia sui gates che sulle matrici di multi - head attention e proiezione; da un report generato durante il training: su circa 3.8 miliardi di parametri ne alleniamo ≈ 9 milioni e tutti i risultati quantitativi che mostriamo sono rispetto ad un training dello 0.23% dei parametri totali.

4.3 Risultati

Riportiamo ora i risultati del fine-tuning con LoRA. La Tabella 6 confronta diverse strategie variando tre iperparametri chiave:

- λ_C: il peso del task di rewriting nella loss totale (dove $\lambda_C = 0$ indica il training single-task).
- w_{pos}: il peso assegnato alla classe positiva (Sarcastic) per bilanciare la loss.
- τ: la soglia applicata al differenziale dei logit ($A - B$) per la classificazione.

Table 6: Phi-3 Fine - Tuning Results)

	Hyperparameters			**Metrics**		
Configuration	λ_C	w_{pos}	τ	**F1**	**Prec.**	**Rec.**
Multitask (Halved C)	0.5	3.0	0.25	0.524	0.420	0.695
Multitask (Best)	1.0	–	0.00	**0.578**	**0.603**	0.555
Multitask (Equal C)	1.0	3.0	0.25	0.511	0.394	0.725
Single Task (A only)	–	3.0	0.00	0.499	0.373	**0.755**

Si osserva come la configurazione *Multitask (Standard)* raggiunga il miglior risultato complessivo (F1 0.578). Il task di rewriting impone un chiaro trade-off: a fronte di una riduzione della Recall (-20 punti percentuali rispetto al Single Task), si ottiene un incremento superiore della Precisione (+23 punti). Questo scambio è vantaggioso poiché abbatte drasticamente i falsi positivi e riduce il problema iniziale che hanno i LLM di considerare, in questo dominio, praticamente sempre sarcastico il testo stesso, rendendo la predizione più affidabile rispetto al Single Task pesato ($w_{pos} = 3.0$), il quale massimizza la Recall (0.755) al prezzo di una Precisione molto bassa e di un F1 score inferiore.
Confrontando **A only** vs **A + C**, si vede che Task C è utile: anche quando il recall rimane alto, l'aggiunta del rewriting riduce il rumore e migliora la qualità del segnale (F1 passa da 0.499 a 0.578 nel best setup). L'effetto del peso relativo di C è coerente a quello che ci aspettiamo e che abbiamo visto anche in ModernBERT: il task ausiliario aiuta le performance complessive, permette un training più continuativo e stabile ma deve essere sempre chiaro il peso maggiore del main task, infatti quando C pesa troppo (C = A con `pos_weight`=3), il modello diventa più "sensibile" (recall 0.725) ma perde selettività (precision 0.394), aumentando i falsi positivi.

4.4 Positive Weight Drift

Il calo di performance osservato introducendo il `pos_weight` su Phi-3, in contrasto con il beneficio ottenuto su BERT, è giustificabile analizzando la diversa

struttura delle architetture e, soprattutto, di come trattiamo la head. In ModernBERT l'ultimo layer è inizializzato casualmente e deve apprendere da zero un decision boundary. Qui, `pos_weight`, come già spiegato, agisce proprio per controbilanciare la spinta della classe maggioritaria e permettendo al modello di convergere su una soglia decisionale equilibrata; il concetto fondamentale è che, qui, la head l'alleniamo da zero ed il ragionamento per Phi è diverso proprio per questo. In Phi-3 il fine-tuning non costruisce un classificatore da zero, ma adatta una distribuzione di probabilità prior relativa al pre - training del modello stesso. Il termine di `pos_weight`, in questo caso, peggiora le perfomance di Phi perché lo rende molto più tendente a sovrastimare la classe positiva "A" che ad imparare il segnale sarcastico a cui puntiamo. La maniera più semplice per spiegarla fa proprio riferimento alla necessità di apprendere, o meno, una scala di valori confrontabili per le classi: in BERT `pos_weight` è importante per 'trovare' la classe minoritaria, in Phi 'crea confusione' con la distribuzione prior delle etichette già esistente, non avendo appunto necessità di dover bilanciare da zero la distribuzione appresa dalla head perché non la alleniamo. Questo non vuol dire che Phi sia completamente esente da qualunque probabilità di prevedere sempre la classe più frequente e difatto non imparare, per questo lo abbiamo provato ma i risultati si sono rivelati molto migliori senza `pos_weight` che con e quella precedente è una possibile spiegazione semplice.

5 Conclusioni

Otteniamo i migliori risultati (di gran lunga) con un fine tuning di Phi-3 con task ausiliario e `pos_weight=0`, superando la baseline iniziale di entrambi i LLM stavolta, sia LLama 70b che Kimi - K2. Il grafico seguente riassume i risultati del nostro lavoro:

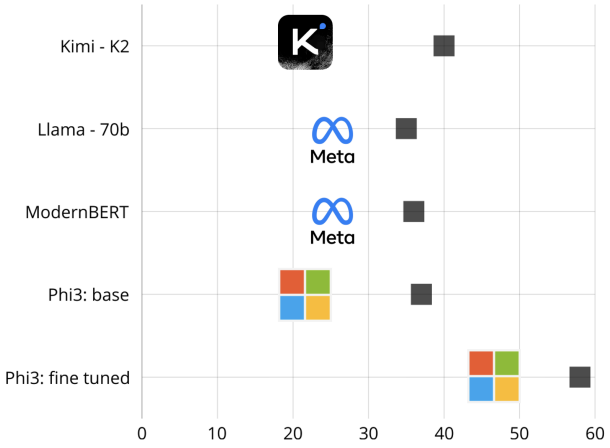

Figure 1: Risultati Best Models and Baselines

Llama-70B mostra una tendenza a classificare come sarcastici testi brevi e tipici del lessico 'social', anche quando il contesto successivo rende l'intenzione dell'autore chiaramente non sarcastica; l'evidenza contestuale ha un impatto limitato sulla della decisione presa dal modello, differentemente da quanto avviene per Kimi e Phi3. Proprio quest'ultimo, immaginiamo, che essendo stato allenato su dataset molto meno grandi e rumorosi (si parla di text quality data da libri), probabilmente eviti di avere questo bias a ritenere sarcastici testi brevi e piccoli, proprio perché non è allenato su dati da Reddit, Twitter... e social vari, dimostrandosi una scelta azzeccata in questo caso. Non è necessariamente vero che anche con il fine tuning, su testi molto lunghi e complessi, allora Phi3 riconosca meglio il sarcasmo di di LLama o Kimi - K2, ma lo fa in questo caso specifico.