UNIVERSITÀ DI PISA

INFORMATICA UMANISTICA

LINGUISTICA COMPUTAZIONALE II

IronITA - Italian irony detection in Twitter

Salvatore Ergoli (660527)

Indice

1 Introduzione

Il *task* affrontato nel seguente progetto è quello della rilevazione dell'ironia nei testi (proposto da EVALITA[1] nel 2018), un tema rilevante per la comprensione delle possibili sfumature del linguaggio naturale. L'obiettivo primario è lo sviluppo di modelli di classificazione capaci di distinguere tra tweet ironici e non ironici. Per raggiungere questo obiettivo sono stati esplorati due diversi approcci: un classificatore con SVM lineari (con differenti rappresentazioni di *features*) e un *neural language model* (che invece lavora su semplice testo grezzo). La relazione documenta l'intero processo: dall'analisi del dataset utilizzato, alle fasi di pre-elaborazione dei dati e alle varie strategie di rappresentazione del testo. Infine, sono presentati i risultati degli esperimenti condotti sul test set insieme alle metriche di valutazione utilizzate per valutare le prestazioni dei vari modelli.

2 Dataset

Il dataset rilasciato da EVALITA è composto da tweet provenienti da diversi corpora [3]: Hate Speech Corpus (Sanguinetti et al., 2018) (HSC) e Twittirò Corpus (Cignarella et al., 2018) (TW-BS), che a sua volta include tweet raccolti da altri corpora come LaBuonaScuola corpus (TWBS) (Stranisci et al., 2016), Sentipolc corpus (TWSENTIPOLC), Spinoza corpus (TW-SPINO) (Barbieri et al., 2016). Nel test set le fonti sono sostanzialmente le stesse, con l'aggiunta di alcuni tweet annotati durante un task di SENTIPOLC 2016 (Barbieri et al., 2016).

Dataset	Ironico	Non ironico	Totale
Training set	2023	1954	3977
Test set	435	437	872

Tabella 1: Distribuzione di frequenza dell'ironia nei due dataset

I dati sono strutturati in formato csv (con il punto e virgola come separatore) e comprendono le seguenti features per ogni record:

1. **id**: un identificatore numerico univoco per ciascun tweet;
2. **text**: il testo del tweet;
3. **irony**: etichetta binaria che può assumere come possibili valori 1 (tweet ironico) e 0 (tweet non ironico);
4. **sarcasm**: altra classe binaria, che non viene utilizzata poiché si è deciso di concentrarsi esclusivamente sul task di Irony Detection;
5. **topic**: corpus di provenienza del tweet (es: HSC).

3 Modelli di classificazione

Durante la fase di classificazione sono stati implementati quattro diversi modelli, per ognuno dei quali è stata riportata la rispettiva tabella di contingenza con le varie metriche (precision, recall, accuracy e F-1 score) Come baseline di riferimento è stato utilizzato un *Dummy Classifier* con la versione *most-frequent*, nella quale tutti i tweet sono etichettati con la classe più frequente. Nel caso specifico, essendo i tweet sul test set quasi perfettamente bilanciati (come si può osservare dalla tabella 1, i tweet non ironici sono solo due in più), l'accuracy è stata del 50%. Per i primi tre modelli il focus non è stato tanto sul classificatore - sempre un SVM lineare con parametro dual impostato a False - ma sull'ingegnerizzazione delle feature, in cui la scelta principale è sul decidere quali caratteristiche pesare per la rappresentazione [8]. Questo ha aiutare a capire quali fossero le migliori features per la risoluzione di un particolare task, cosa invece non più possibile con il modello pre-addestrato di Bert (il quarto e ultimo), in cui le rappresentazioni sono invece delle *black box*. In sintesi, i modelli utilizzati sono i seguenti:

- Classificatore svm lineare con feature linguistiche estratte da Profiling-UD[2];

[1] https://www.evalita.it/
[2] http://www.italianlp.it/demo/profiling-ud/

- Classificatore svm lineare con feature lessicali (n-grammi di caratteri, parole, lemmi e Part-of-Speech);
- Classificatore svm lineare con word embedding estratti dall'italiaNLP Lab[3];
- Neural Language Model (Bert e Distil-BERT).

3.1 SVM con features linguistiche non lessicali

Il primo classificatore è stato implementato utilizzando il sistema Profiling-UD per estrarre informazioni linguistiche non lessicali dai testi. Questo tool fornisce una vasta gamma di feature linguistiche, che includono oltre 130 rappresentazioni della struttura lessicale, morfologica e sintattica del singolo tweet. Le feature linguistiche vengono ottenute attraverso un'iniziale annotazione automatica dei testi da parte di UDPipe[4], una pipeline che esegue la prima fase di pre-processing del testo (sentence splitting, tokenizzazione, POS Tagging, lemmatizzazione e parsing sintattico). In generale, i fenomeni linguistici descritti possono essere raggrupati in 7 categorie [1]:

1. **Proprietà del testo grezzo:** lunghezza del documento, lunghezza media delle frasi, lunghezza media delle parole;
2. **Varietà lessicale:** Type Token Ratio (TTR)[5];
3. **Informazione morfosintattica:**: distribuzione delle categorie grammaticali, densità lessicale[6], morfologia flessionale;
4. **Struttura del predicato verbale:** distribuzione delle teste verbali, distribuzione delle radici verbali;
5. **Strutture globali e locali dell'albero di parsing:** profondità media dell'albero sintattico, lunghezza media delle proposizioni, lunghezza delle relazioni di dipendenza;
6. **Relazioni sintattiche:** distribuzioni delle relazioni di dipendenza;
7. **Uso della subordinazione:** distribuzione delle subordinate e delle proposizioni principali, ordine relativo delle subordinate rispetto alla testa verbale.

Tenendo conto che per l'analisi sono stati presi in considerazioni singoli tweet, i vettori ottenuti dal file .csv di Profiling-UD sono vettori sparsi, in quanto la lunghezza di un tweet è limitata e quindi incapace di cogliere tutte le possibili rappresentazioni linguistiche descritte sopra. Per tale motivo, si è deciso di eliminare tutte le feature con un numero di zeri superiore a una soglia del 90%: in questo modo si è ridotto lo spazio delle feature da 133 ad 82. A queste 82 feature, ne sono state aggiunte altre 5, prendendo spunto dalle linee guida dettate dagli autori[2] del Twittirò Corpus per etichettare un tweet come ironico. Infatti, oltre a fenomeni più difficili da riconoscere, come analogie, eufemismi, paradossi e iperboli, vengono elencati anche una serie di *trigger* sintattici e semantici (ad esempio segni di punteggiatura, parole negative ed entità nominali) molto più facili da identificare e categorizzare.

Pertanto, per questa prima analisi di tipo non lessicale, sono state aggiunte 5 nuove feature:

- *exclamation_count*: numero di punti esclamativi;
- *question_count*: numero di punti interrogativi;
- *dot_count*: numero di punti;
- *negative_word*: numero di 'non' all'interno di un tweet;
- *custom_features*: la somma cumulata delle precedenti quattro feature inserite.

[3]http://www.italianlp.it/resources/italian-word-embeddings/
[4]https://ufal.mff.cuni.cz/udpipe/1/models
[5]Misura lessicale che è il rapporto tra il numero di parole tipo e di parole token.
[6]Rapporto tra il numero di parole contenuto e il numero di parole totali nel testo.

Necessario, prima dell'addestramento del modello, è stato normalizzare ciascun vettore di feature mediante *MinMaxScaler*[7], una tecnica che uniforma tutti i valori in un intervallo compreso tra 0 e 1. Successivamente, il modello è stato valutato attraverso un processo di 5-fold cross-validation, un approccio che comporta la suddivisione del training set in cinque fold distinti di cui - in maniera iterativa - quattro vengono utilizzati come training e uno come test. Questa metodologia permette di ottenere una valutazione maggiormente affidabile (rispetto all'utilizzo del solo set di addestramento) delle prestazioni del modello su dati precedentemente non visti. I risultati di ciascun fold generato, comparato alla rispettiva baseline, vengono riportati nella tabella sottostante:

Fold	**Accuracy**	**Baseline**
Fold-1	0.64	0.50
Fold-2	0.66	0.49
Fold-3	0.65	0.50
Fold-4	0.63	0.53
Fold-5	0.62	0.50

Tabella 2: Valutazione del sistema con una 5-fold cross validation sul training set

Infine, nella tabella 3 vengono riportate le performance sul test set:

Classi	Misure			
	Precision	Recall	F-1 score	Accuracy
0	0.60	0.72	0.65	0.62
1	0.65	0.51	0.57	

Tabella 3: Misure di valutazione del modello

Mostrando la feature importance per la risoluzione del task (figura 1), si nota che la seconda feature più importante è *question_count*, creata manualmente durante la fase di pre-processing. Questo può essere spiegato dal fatto che, quando scriviamo una domanda siamo più propensi a usare il sarcasmo o l'ironia, in quanto la domanda stessa crea una certa aspettativa e un'ambiguità che può essere sfruttata per aggiungere una sfumatura ironica.

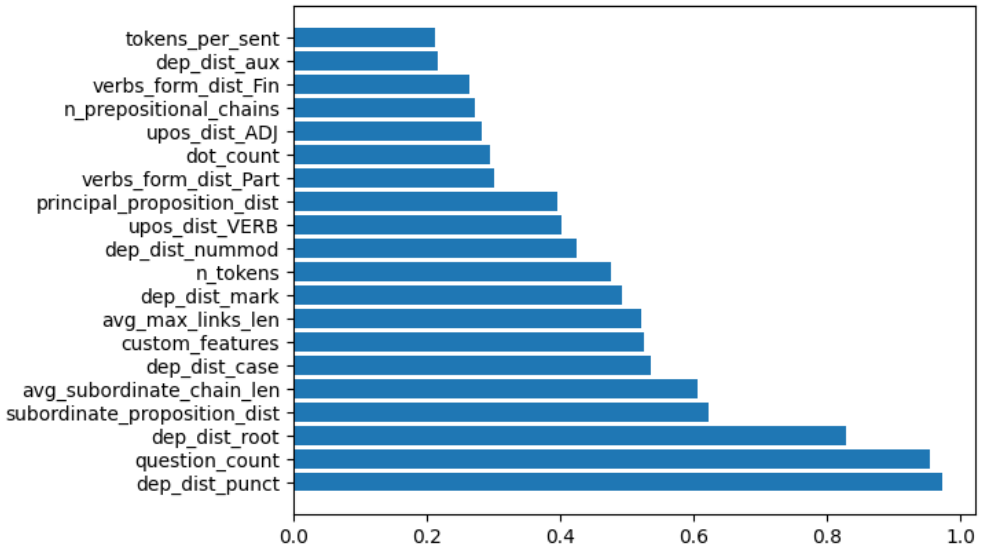


Figura 1: Elenco delle 15 features più importanti per la classificazione

[7]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

3.2 SVM con features lessicali

Nel secondo classificatore ad essere rappresentate come features sono n-grammi di parole, caratteri, lemmi e part-of-speech (POS). In particolare, l'utilizzo di n-grammi di caratteri ha aiutato a garantire che anche parole diverse, ma con una sottoparte uguale, fossero pesate in maniera simile durante il processo di addestramento. Ciò ha contribuito a migliorare la capacità del modello di generalizzare su parole simili ma non identiche, mitigando il problema della *data sparseness*[5]. Per provare a rendere ancora più granulare il modello, si è deciso di eliminare le *stop words*[8], parole molto frequenti e poco informative. In tal modo, si è ridotta ulteriormente la dimensionalità del vettore di features.

Com'è possibile osservare dalla tabella 4, la miglior accuracy media con cross validation uguale a 5 è stata raggiunta dai classificatori di unigrammi, bigrammi e trigrammi di parole, combinati a bigrammi e trigrammi di caratteri. Risultati inferiori sono stati raggiunti dai modelli che facevano uso di n-grammi di POS, questo probabilmente perché l'ironia si basa su elementi lessicali e semantici che vanno oltre la struttura di base del testo.

N°	Modello	**Accuracy media**
1	Classificatore con unigrammi, bigrammi e trigrammi di parole e bigrammi e trigrammi di caratteri	0.69
2	Classificatore con unigrammi, bigrammi e trigrammi di parole, unigrammi, bigrammi e trigrammi di pos	0.66
3	Classificatore con unigrammi, bigrammi e trigrammi di pos e bigrammi e trigrammi di caratteri	0.67
4	Classificatore unigrammi, bigrammi e trigrammi di pos e bigrammi e trigrammi di lemmi	0.63
5	Classificatore con unigrammi, bigrammi e trigrammi di lemmi e bigrammi e trigrammi di caratteri	0.68

Tabella 4: Accuracy media della 5-fold cross validation per ogni modello

Di seguito la 5-fold cross validation per il modello 1, ossia il migliore a performare sul training set:

Fold	**Accuracy**
Fold-1	0.67
Fold-2	0.69
Fold-3	0.69
Fold-4	0.68
Fold-5	0.68

Tabella 5: Valutazione del sistema con una 5-fold cross validation sul training set

Confrontando i risultati del modello SVM con features lessicali (n-grammi di parole e caratteri) con quelli del modello SVM con features linguistiche estratte da Profiling-UD, si evidenzia un miglioramento significativo nell'accuratezza, che passa dal 62% al 66%.

Classi	Misure			
	Precision	Recall	F-1 score	Accuracy
0	0.68	0.60	0.64	0.66
1	0.64	0.71	0.67	

Tabella 6: Misure di valutazione del modello

[8]Le stop words utilizzate in questo progetto sono state prelevate dalla seguente repository GitHub: https://github.com/stopwords-iso/stopwords-it/tree/master

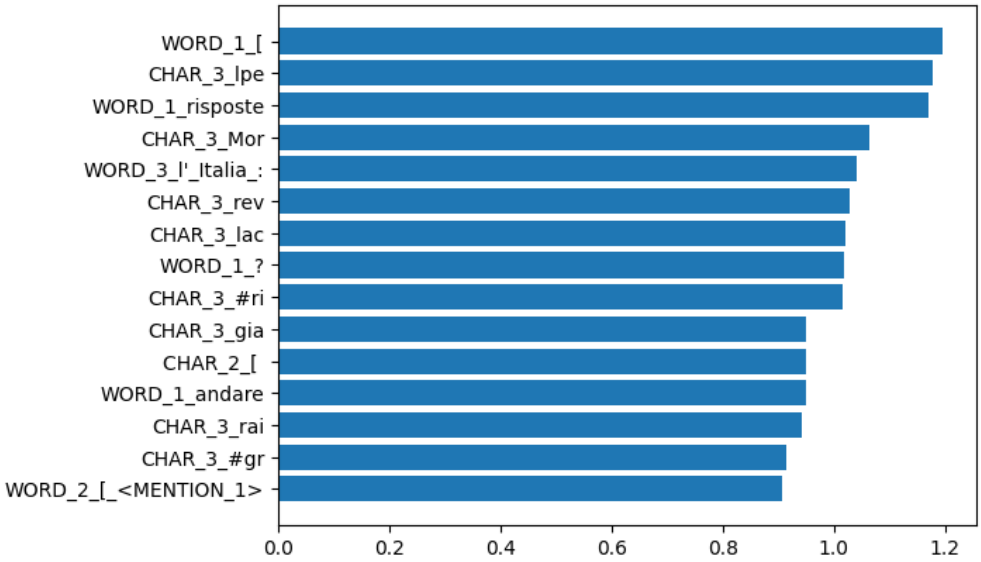


Figura 2: Elenco delle 15 features più importanti per la classificazione

3.3 SVM con word embeddings

Il terzo approccio è quello dei *word embeddings*, in cui abbiamo una rappresentazione che non riguarda più le proprietà intrinseche della parola (n-grammi di caratteri, lemma, part of speech) o del tweet (features linguistiche estratte da Profiling-UD), ma lo spazio semantico in cui è racchiusa la parola stessa. In altri termini: il suo contesto[9]. In questo contesto, le parole sono codificate in vettori multidimensionali, consentendo di catturare le loro caratteristiche semantiche in una matrice densa, contrapposta alle matrici sparse dei modelli precedenti. Per il task in questione, sono stati utilizzati gli embedding riguardanti la sfera di Twitter prodotti dall'Italian-NLP Lab[10]. Per ragioni di costo computazionale, si è deciso di scegliere gli embeddings codificati in 64 dimensioni.

Prima di utilizzare gli embeddings per addestrare l'SVM, è necessaria una fase di pre-elaborazione dei tweet per provare a migliorare la qualità degli embedding, rendendoli più accurati e informativi.

Il testo è stato normalizzato nel seguente modo:

1. **Sostituzione dei numeri**: i numeri sono stati sostituiti con il simbolo '@Dg' se erano maggiori di 2100;
2. **Normalizzazione delle lunghezze delle parole**: le parole con più di 26 caratteri sono state sostituite con la stringa "LONG-LONG". Questo è stato fatto per evitare problemi di rappresentazione di parole molto lunghe negli embedding;
3. **Normalizzazione del case**: le parole sono state convertite in minuscolo, ma la prima lettera della prima parola in una frase - se in maiuscolo - non è stata alterata. Questo ha contribuito a mantenere la struttura sintattica del testo e a evitare che il modello interpretasse parole con la stessa ortografia ma *case* diverso come parole distinte.

Per mettere insieme gli embedding di tutte le parole di un tweet e creare una rappresentazione vettoriale univoca, sono stati utilizzati diversi approcci:

- **Media semplice**: la media vettoriale di tutti gli embedding delle parole nel tweet;

[9] *"You shall know a word by the company it keeps"*, Questo citazione, di John Rupert Firth, riassume l'importanza cruciale del contesto nell'interpretazione del linguaggio, l'idea alla base della semantica distribuzionale

[10] http://www.italianlp.it/resources/italian-word-embeddings/

- **Media con normalizzazione L1**: la media vettoriale degli embedding delle parole nel tweet, dopo aver normalizzato ogni embedding con la norma L1[11].;
- **Media per categorie lessicali:** la media degli embedding di tutte le parole che hanno la stessa categoria lessicale (ad esempio, tutti i nomi, tutti i verbi, etc...);
- **Concatenazione di medie per categorie lessicali**: la media degli embedding (per le categorie lessicali scelte) viene calcolata separatamente e, solo alla fine, le medie di ogni categoria vengono concatenate in unico vettore che rappresenta l'intero tweet.

N°	Modello	**Accuracy media**
1	Classificatore con media semplice	0.70
2	Classificatore con normalizzazione L1	0.68
3	Classificatore con media per categorie lessicali	0.66
4	Classificatore con concatenazione di medie per categorie lessicali	0.66

Tabella 7: Valutazione del sistema con una 5-fold cross validation sul training set

Nonostante il tentativo di provare ad usare embedding maggiormente raffinati, i migliori risultati sono stati ottenuti facendo una media semplice tra tutte le parole di ciascun tweet. Questo è in linea con gli attuali trend del NLP, in cui i modelli additivi (quelli, per l'appunto, realizzati con una semplice somma mediata) restano competitivi al netto dell'utilizzo di architetture più sofisticate come RNN, LSTM e Trasformers per la creazione degli embedding [6].

Come per il precedente modello di features lessicali, viene mostrata la 5-fold cross validation per il modello con la miglior accuracy sul training set:

Fold	**Accuracy**
Fold-1	0.71
Fold-2	0.70
Fold-3	0.71
Fold-4	0.69
Fold-5	0.71

Tabella 8: Valutazione del sistema con una 5-fold cross validation sul training set

Classi	Misure			
	Precision	Recall	F-1 score	Accuracy
0	0.69	0.56	0.62	0.65
1	0.63	0.75	0.68	

Tabella 9: Misure di valutazione del modello

3.4 Neural Language Model

Nel quarto e ultimo modello, invece di ingegnerizzare manualmente le features, si è sfruttato un modello linguistico pre-addestrato: **BERT** (Bidirectional Encoder Representations from Transformers) [4], che ha dimostrato eccellenti prestazioni in molti task di elaborazione del linguaggio naturale, tra cui la comprensione del testo e la classificazione.

[11] La norma L1 di un vettore è definita come la somma dei valori assoluti delle sue componenti. Quindi, per un vettore $\mathbf{v} = (v_1, v_2, \ldots, v_n)$, la norma L1 è $\|\mathbf{v}\|_1 = \sum_{i=1}^{n} |v_i|$.

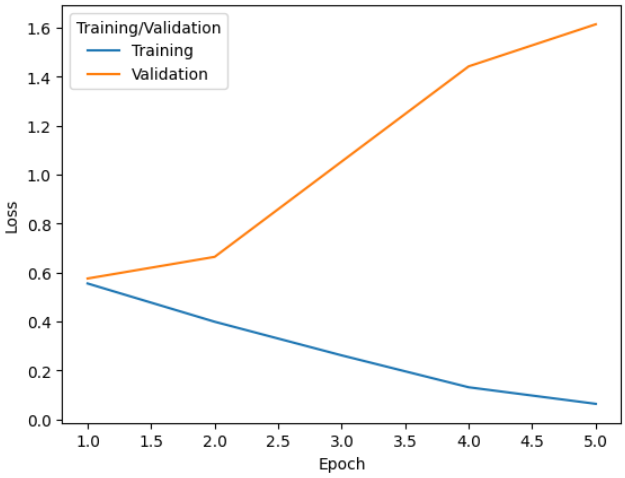
Per il task sull'ironia, sono stati utilizzati due modelli BERT pre-addestrati per la lingua italiana:

- **Italian-BERT-base-cased:** modello BERT standard con un numero di parametri pari a 125 milioni;
- **Italian-DistilBERT-base-cased**: versione distillata[12] di Italian-BERT-base-cased con un numero di parametri pari a 66 milioni.

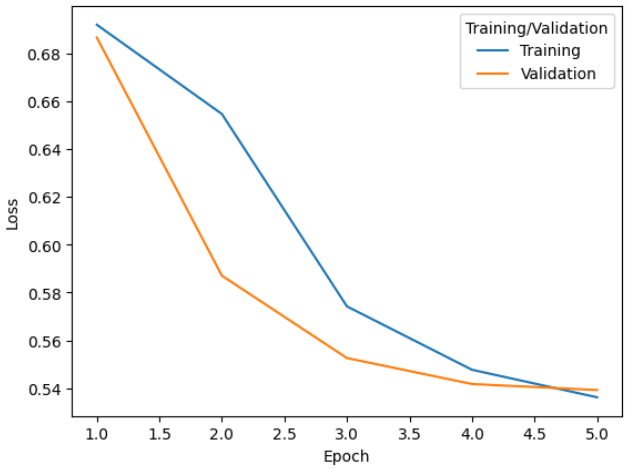
L'utilizzo di entrambi i modelli ha permesso di analizzare il trade-off tra accuratezza e costo computazionale. In particolare, si è voluto valutare se le prestazioni migliori del modello con maggiori parametri giustifichino il maggiore costo computazionale o se DistilBERT [9] offra un buon compromesso tra prestazioni ed efficienza.

Per entrambi i modelli BERT, l'input è costituito dal testo grezzo del tweet, senza ulteriori pre-elaborazioni; il processo di fine-tuning è avvenuto per 5 epoche, con un learning rate di 2e-6, un weight decay di 0.01 e un batch size di 16.

Com'è possibile osservare nella figura 3a, mentre sul training la loss di Bert diminuisce costantamente, sul validation la loss aumenta in maniera significativa ad ogni epoca. Quest'andamento delle curve suggerisce un problema di overfitting, abbastanza plausibile considerate le dimensioni del dataset a disposizione: il modello - che è molto complesso - si adatta troppo bene ai dati di training ma non generalizza bene su quelli di validation. In DistilBERT invece (3b) la loss diminuisce costantamente con il passare delle epoche sia su training che su validation. Questo suggerisce che DistilBERT non soffre di overfitting e generalizza meglio rispetto a Bert.



(a) Learning curve di Bert-base



(b) Learning curve di DistilBERT

Figura 3: Learning curves dei modelli Bert

I risultati sul test set mostrano che entrambi i modelli raggiungono performance migliori rispetto alle svm lineari, com'era lecito aspettarsi. DistilBERT, con un'accuracy complessiva di 0.70, presenta una recall più alta rispetto a BERT (0.75 vs 0.66) sulla classe 1, ma un F-1 score inferiore (0.70 vs 0.72). Questa tendenza suggerisce che DistilBERT riesca a identificare un numero maggiore di tweet ironici rispetto a BERT, ma che, in alcuni casi, sia meno preciso nel classificarli correttamente, con conseguente minor F-1 score e accuratezza complessiva.

[12]La distillazione è una tecnica di compressione dei modelli di apprendimento automatico che addestra un modello "studente" più piccolo ad imitare il comportamento di un modello "insegnante" più grande, preservando la maggior parte della sua conoscenza, ma con dimensioni e complessità ridotte.

Classi	Misure			
	Precision	Recall	F-1 score	Accuracy
0	0.73	0.65	0.69	0.70
1	0.68	0.75	0.72	

Tabella 10: Misure di valutazione di DistilBERT

Classi	Misure			
	Precision	Recall	F-1 score	Accuracy
0	0.70	0.77	0.73	0.72
1	0.74	0.66	0.70	

Tabella 11: Misure di valutazione di Bert

L'aspetto interessante è che DistilBERT, a discapito del minor numero di parametri con i quali è stato addestrato (circa la metà rispetto a Bert), raggiunga prestazioni quasi equivalenti, a conferma del fatto che questo modello più leggero potrebbe essere una scelta valida anche per scenari con risorse limitate o per dataset piccoli maggiormente proni all'overfitting.

4 Conclusioni

I risultati ottenuti evidenziano una delle sfide più grandi nella rilevazione dell'ironia: la dissonanza tra forma superficiale e sottotesto. I modelli linguistici, come i classificatori lineari con SVM, per la classificazione si sono affidati all'analisi della forma superficiale del testo, focalizzandosi sulla struttura sintattica, sulla distribuzione lessicale e su indizi linguistici espliciti. Questa analisi è in grado di cogliere segnali di ironia diretti e evidenti, ma risulta inefficace nel decifrare l'ironia sottile, che si nasconde in un sottotesto che si alimenta di inferenze contestuali, giochi di parole e ambiguità.

Le ricerche condotte hanno dimostrato il vantaggio dei modelli linguistici pre-addestrati, come BERT e DistilBERT, che tuttavia continuano ad avere dei limiti nella corretta interpretazione dell'ironia. Per questo motivo, la sfida futura è quella di sviluppare modelli che possano andare oltre la forma superficiale, spingendosi nella dimensione più profonda del significato. Questo significa investire sulla comprensione del contesto, sulla codifica della conoscenza umana, sulla capacità di analizzare la multimodalità della comunicazione e sulla creazione di sistemi di apprendimento che non siano limitati a trattare il testo come una serie di parole, ma come un contenitore di significato e intenzioni [7].

Riferimenti bibliografici

[1] Dominique Brunato, Andrea Cimino, Felice Dell'Orletta, Giulia Venturi, and Simonetta Montemagni. Profiling-ud: a tool for linguistic profiling of texts. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 7145–7151, 2020.

[2] Alessandra Teresa Cignarella, Cristina Bosco, Viviana Patti, and Mirko Lai. Application and analysis of a multi-layered scheme for irony on the italian twitter corpus twittirò. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[3] Alessandra Teresa Cignarella, Simona Frenda, Valerio Basile, Cristina Bosco, Viviana Patti, Paolo Rosso, et al. Overview of the evalita 2018 task on irony detection in italian tweets (ironita). In *CEUR Workshop Proceedings*, volume 2263, pages 1–6. CEUR-WS, 2018.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[5] A. Lenci, S. Montemagni, and V. Pirrelli. *Testo e computer: elementi di linguistica computazionale*. Aula magna. Carocci, 2016.

[6] A. Lenci and M. Sahlgren. *Distributional Semantics*. Studies in Natural Language Processing. Cambridge University Press, 2023.

[7] Alessandro Lenci. Understanding natural language understanding systems. a critical analysis, 2023.

[8] Zhiyuan Liu, Yankai Lin, and Maosong Sun. *Representation learning for natural language processing*. Springer Nature, 2023.

[9] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.