UNIVERSITÀ DI PISA

Progetto di Linguistica Computazionale II:
Valutazione di modelli computazionali per la classificazione dell'ironia

Progetto B

Autore: Alice Bergonzini
Matricola: 560680

9 luglio 2023

1 Introduzione

La presente relazione descrive il progetto svolto per l'esame del corso di Linguistica Computazionale II il quale consiste nello sviluppo, la valutazione e il test di diversi modelli per un task di classificazione linguistica specifico: la classificazione dell'ironia. Il dataset utilizzato per questo scopo è quello messo a disposizione da EvalIta per lo shared task IronIta del 2018 sulla rilevazione dell'ironia[1].
L'obiettivo principale di questo progetto è valutare e confrontare diverse strategie di machine learning per la classificazione binaria. Per raggiungere questo obiettivo, sono stati sviluppati e testati diversi modelli: in particolare, sono stati sviluppati diversi modelli di Support Vector Machine lineari e infine, è stato fatto del fine-tuning su un modello pre-addestrato di tipo transformer. I modelli sono stati addestrati utilizzando tecniche di apprendimento supervisionato.
La relazione fornirà una panoramica dei vari modelli sviluppati, delle strategie implementate e dei risultati ottenuti. Si illustreranno i diversi modelli in due parti diverse: la prima tratterà delle strategie di input impiegate con lo stesso SVM lineare, mentre nella seconda si parlerà del processo di fine-tuning svolto sul modello pre-addestrato.

[1] http://www.di.unito.it/ tutreeb/ironita-evalita18/index.html

2 Descrizione del task

Il task selezionato per questo progetto è la rilevazione dell'ironia, che fa parte dello shared task proposto da EvalIta nel 2018. In questa sezione verranno presentati il task stesso e il dataset fornito per lo sviluppo del progetto.
La rilevazione dell'ironia rappresenta una sfida nell'ambito dell'elaborazione del linguaggio naturale, poiché richiede la comprensione del contesto, delle sfumature linguistiche e delle intenzioni dell'autore [Cignarella et al., 2018].

2.1 Obiettivi e scelta del task

EvalIta propone due sottotask per questo dataset[Cignarella et al., 2018]:

- *Rilevazione dell'ironia:* Consiste in una classificazione binaria in cui i sistemi devono prevedere se un tweet è ironico o meno.
- *Classificazione di diversi tipi di ironia:* si focalizza sulla rilevazione di diversi tipi di ironia, con particolare attenzione all'identificazione del sarcasmo. In particolare, si tratta di una classificazione multi-classe, in cui i sistemi devono predire una delle tre seguenti etichette: i) sarcasmo, ii) ironia non categorizzata come sarcasmo e iii) non-ironia.

Nel contesto di questo progetto, ho scelto di concentrarmi sul sotto-task della rilevazione dell'ironia. Ho sviluppato, addestrato, validato e testato diversi modelli con l'obiettivo di ottenere uno strumento per la classificazione binaria dell'ironia automatica. Lo scopo principale è stato quello di valutare le varie strategie, a partire dalla rappresentazione dell'input, per affrontare con successo questo compito.

2.2 Dataset

Il dataset di IronIta utilizzato per questo progetto è composto da un totale di 3977 esempi per il training e 872 esempi per il test. Ogni esempio rappresenta un tweet e riporta le seguenti informazioni per ciascuno di essi:

- *Id:* Un identificativo univoco
- *Text:* Il testo del tweet
- *Irony:* Annotazione rispetto alla categoria dell'ironia espressa in valori booleani (0 e 1)
- *Sarcasm:* Annotazione rispetto alla categoria del sarcasmo espressa in valori booleani (0 e 1)
- *Topic:* Indica il corpus di provenienza del tweet

Per questo progetto, le informazioni relative all'ID e al topic dei tweet non sono state considerate rilevanti per l'allenamento dei modelli. Inoltre, l'annotazione del sarcasmo, che è specifica del secondo sotto-task, non è stata utilizzata come dato durante il processo di addestramento. Al contrario, l'annotazione dell'ironia è stata impiegata come annotazione per l'apprendimento supervisionato dei modelli.

3 Implementazione del software

Il codice per lo svolgimento di questo progetto è stato scritto in Python ed è organizzato in notebook Jupyter, ognuno dedicato a un punto specifico del progetto, con dettagli sul pre-processing dei dati e lo sviluppo dei modelli. Nel codice, sono presenti celle di codice commentate per l'implementazione e celle di markdown per descrivere il processo e i risultati. La libreria principale utilizzata per l'addestramento, la validazione e il testing dei modelli SVM è *scikit-learn*.
Per l'ultimo punto del progetto, ovvero quello riguardante il fine-tuning, è stato utilizzato come ambiente di sviluppo un notebook Google Colab, per usufruire delle GPU offerte dalla piattaforma. In questa sezione del progetto come librerie sono state utilizzate principalmente *pytorch* e *transformer*. Il codice, insieme ai dati di input e ai modelli sviluppati, è disponibile nel repository del mio personale GitHub[2].

4 Modelli con SVM lineare

I primi tre modelli sviluppati per la classificazione binaria in questo progetto si basano tutti sui modelli di Support Vector Machine (SVM) lineari. Questi modelli adottano una strategia di apprendimento che mira a trovare l'iperpiano ottimale, ovvero quello con il margine più ampio, per separare i dati nelle rispettive classi[Burges, 1998]. In questa sezione del progetto, vengono confrontati modelli di SVM lineare addestrati su diversi tipi di features. I tipi di rappresentazione presi in considerazione sono i seguenti nell'ordine:

- *Linguistic profiling:* sono state utilizzate features di language-profiling estratte attraverso il tool Profiling-UD.
- *N-grammi:* sono stati estratti n-grammi di diversi tipi a partire dalle annotazioni dei tweet.
- *Word Embedding:* si utilizzano i word embedding come features linguistiche.

Per ciascuna delle rappresentazioni all'interno di una specifica sotto-sezione, vengono forniti dettagli sul flusso di lavoro seguito e i risultati ottenuti. In queste sotto-sezioni viene descritto come i dati sono stati pre-processati per l'addestramento del modello SVM lineare, inclusa l'estrazione delle features specifiche per la rappresentazione considerata. Successivamente, vengono riportati i risultati ottenuti in termini di accuratezza e altre metriche di valutazione per valutare le prestazioni dei modelli su ciascuna rappresentazione dei dati.

4.1 Features di Linguistic Profiling

Per il primo punto del progetto, è stato sviluppato un modello di SVM lineare utilizzando features di profilazione linguistica estratte con il tool Profiling-UD. Queste features rappresentano una serie di caratteristiche del testo basate sulla sua forma, come per esempio proprietà del testo grezzo (e.g lunghezza delle frasi, lunghezza delle parole) oppure proprietà morfo-sintattiche (e.g, distribuzione di categorie grammaticali). Considerando che Profiling-UD è uno strumento che crea le premesse per studi che

[2]https://github.com/alicebergonzini/LCII_project

si concentrano soprattutto sulla forma piuttosto che sul contenuto di un testo, possiamo aspettarci risultati poco soddisfacenti per un compito come la rilevazione dell'ironia [Brunato et al., 2020].

4.1.1 Workflow

Al fine di estrarre le caratteristiche utilizzando il tool di profiling UD, è stato necessario adattare il dataset di IronIta al formato richiesto per l'elaborazione dello strumento. A tal scopo, per ogni esempio presente nel file csv di IronIta, è stato creato un file txt che contenesse le informazioni principali nel nome del file e il testo del tweet nel corpo del file. Questi file sono stati compressi in un file zip e successivamente utilizzati come input per il tool di Profiling-UD, ottenendo così un file csv con tutte le caratteristiche di profilazione.
In secondo luogo, è stato eseguito un processo di pre-processing per preparare il dataset in modo da renderlo adatto all'addestramento del modello SVM di *scikit-learn*. Per questo è stato necessario trasformare le features per i singoli esempi in vettori e fare lo scaling dei valori.
Come passaggio successivo, è stato eseguito un processo di K-Fold Validation sul modello. In seguito, il modello è stato testato utilizzando il test set ufficiale di IronIta. Infine, sono state estratte le 15 caratteristiche più rilevanti per la classificazione. I risultati di queste valutazioni sono riportati nella sezione successiva.

4.1.2 Risultati

La K-Fold Validation è stata effettuata con 5 fold. L'accuracy per ogni fold è riportata nella tabella 1. La media aritmetica delle accuratezze sulle 5 fold è di 0.62.

Fold	Accuracy
1	0.649
2	0.598
3	0.634
4	0.610
5	0.640

Tabella 1: Score di Accuracy per ogni Fold

Nel test, il modello ha raggiunto un'accuratezza di **0.64**. Come punto di riferimento, è stata utilizzata una baseline che predice semplicemente la classe più frequente, la quale ha ottenuto un'accuratezza di 0.50 nel test. Sulla base di questi risultati, possiamo affermare che il modello SVM addestrato utilizzando il linguistic profiling ha dimostrato di avere una capacità predittiva superiore rispetto alla baseline.

Nelle SVM, l'importanza delle features riveste un ruolo fondamentale per determinare il confine decisionale tra le classi. Nel contesto di questo modello SVM, riportiamo le 15 feature più rilevanti nella figura 1. Al primo posto per importanza troviamo la feature che caratterizza la distribuzione media delle relazioni sintattiche di tipo *discourse*. Se-

guendo la definizione di Universal Dependencies[3], si tratta della relazione che viene utilizzata per le interiezioni e altre particelle e elementi discorsivi che non sono strettamente legati alla struttura della frase, se non in modo espressivo[de Marneffe et al., 2014]. Questo è un punto interessante perché potrebbe suggerire che gli elementi discorsivi svolgono un ruolo cruciale nel contesto della classificazione dell'ironia.

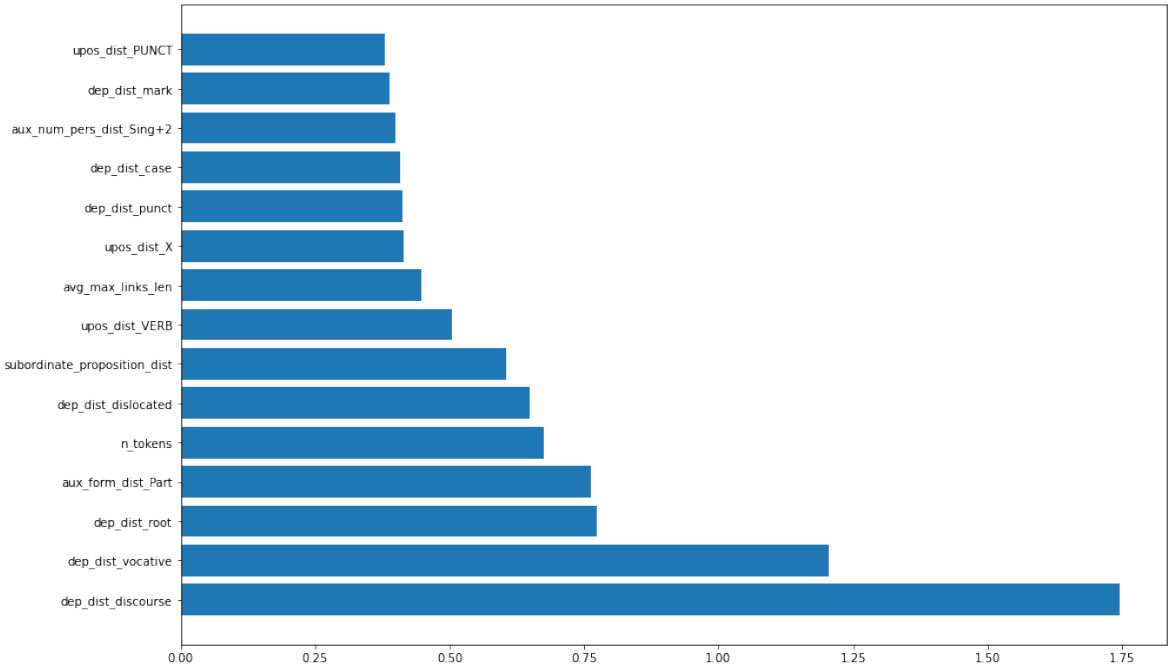

Figura 1: Le 15 feature più importanti per la classificazione da parte del modello

4.2 Features di N-Grammi

Come secondo punto del progetto, si è sviluppato un modello SVM utilizzando gli N-Grammi come input. Gli N-Grammi rappresentano sequenze di n elementi (come parole o caratteri) all'interno di un testo. Le occorrenze di queste sequenze nei singoli esempi di testo, nel nostro caso i tweet, sono state utilizzate come feature per addestrare il modello.
Durante gli esperimenti condotti in questa parte del progetto, sono stati considerati e testati n-grammi di diverse dimensioni e tipologie. Di seguito sono riportati i tipi di n-grammi utilizzati:

- unigrammi, bigrammi e trigrammi di Parole
- unigrammi e bigrammi di Lemmi
- unigrammi e bigrammi di Part of speech
- unigrammi, bigrammi e trigrammi di Caratteri

[3]Definizione della relazione sintattica di tipo discourse dal sito di UD: `https://universaldependencies.org/it/dep/discourse.html`

4.2.1 Workflow

Per l'addestramento del modello SVM è stato necessario creare una matrice di occorrenze degli n-grammi. Per fare questo, la prima parte di pre-processing è stata dedicata all'estrazione degli n-grammi, a partire dai file conllu dei diversi tweet generati tramite profiling-UD. Successivamente, sono state conteggiate le occorrenze degli n-grammi e applicato un filtro per eliminare quelli poco frequenti, al fine di ridurre la dimensionalità della matrice. Infine, è stata generata la matrice di occorrenze degli n-grammi.
Dopodiché per testare la variabilità dell'accuratezza in base al tipo di n-gramma utilizzato come input è stata eseguita una k-fold con diverse combinazioni di n-grammi. Successivamente, al fine di valutare la variazione dell'accuratezza in base al tipo di n-gramma utilizzato come input, è stata eseguita una procedura di k-fold validation utilizzando diverse combinazioni.
Infine, è stato testato sul test set ufficiale di IronIta il modello allenato sulla combinazione di input che ha raggiunto la più alta accuratezza durante la k-fold.

4.2.2 Risultati

Nella tabella 2 si riportano i risultati delle valutazioni per ogni combinazione testata. La combinazione che ha ottenuto l'accuratezza migliore, anche se di poco, è quella composta da unigrammi di caratteri, bigrammi di caratteri, unigrammi di parole e bigrammi di parole.

	Fold 1	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**	**Avg**
char n1; char n2; word n1; word n2;	**0.685**	**0.696**	**0.698**	**0.701**	**0.692**	**0.694**
lemma n1; lemma n2; PoS n1; PoS n2;	0.683	0.672	0.692	0.697	0.682	0.685
char n2; char n3; word n2; word n3;	0.686	0.687	0.680	0.678	0.680	0.682
char n1; word n1; lemma n1; PoS n1;	0.697	0.673	0.687	0.675	0.668	0.680
lemma n1; lemma n2; word n1; word n2;	0.700	0.688	0.678	0.697	0.702	0.693

Tabella 2: Score di accuratezza ottenuti durante la 5 fold per ogni combinazione

Il modello allenato con la combinazione migliore individuata durante la k-fold, testato sul test set ufficiale ha ottenuto un'accuratezza di **0.66**.
Nonostante la semplicità delle SVM e la complessità del compito di classificazione dell'ironia, è stato dimostrato anche in questo caso, che questo modello riesce a fornire previsioni significativamente più accurate rispetto alla baseline

4.3 Features di Word Embeddings

L'ultimo tra gli SVM lineari sviluppati per questo progetto è quello allenato su delle rappresentazioni distribuzionali del testo note come word embedding. Queste rappresentazioni, che a livello concreto prendono la forma di un vettore, sono in grado di caratterizzare a livello semantico i singoli token in base al contesto, posizionando parole con significati simili in punti vicini all'interno dello spazio semantico[Lenci, 2018].
I word embedding utilizzati per questa parte del progetto sono quelli sviluppati per EvalIta dall'ItaliaNLP Lab[Comino et al., 2018]

4.3.1 Workflow

Durante il pre-processing dei dati, è stato come prima cosa svolto un processo di normalizzazione; e poi è stata effettuata l'estrazione degli embedding per ogni tweet. Una volta ottenuti, i singoli embedding dei token, sono stati aggregati e combinati secono strategie diverse. Dopo aver ottenuto i vettori delle features rappresentativi per ogni post, è stato eseguito il processo di K-Fold con 5 iterazioni. Le diverse strategie sono state valutate e successivamente è stata selezionata la migliore per il testing sul test set ufficiale.

4.3.2 Risultati

Sono state testate diverse strategie per l'estrazione e la combinazione dei dati, le quali differiscono su tre aspetti principali:

- *le part of speech prese in considerazione:* tutte o solo parole piene
- *la funzione di aggregazione:* media, somma o prodotto
- *aggregazione degli embedding suddivisi per PoS:* unita o separata

Sono state eseguite validazioni a 5-fold per ciascuna di queste opzioni. Nella Tabella 3 sono riportati i punteggi di accuratezza media delle 5 iterazioni per ogni opzione. Dai risultati, si può osservare che la strategia che ha ottenuto l'accuratezza media più alta è quella che considera tutte le PoS e aggrega gli embedding tramite la somma. Inoltre, poiché non viene effettuata una selezione specifica delle PoS, l'aggregazione avviene in modo globale senza separazione.

PoS	**Aggr F**	**Sep**	**Avg Acc**
all	avg	no	0.659
all	**sum**	**no**	**0.671**
all	prod	no	0.489
adj, verb, noun,	avg	no	0.637
adj, verb, noun,	sum	no	0.634
adj, verb, noun,	prod	no	0.520
adj, verb, noun,	avg	yes	0.620
adj, verb, noun,	sum	yes	0.630
adj, verb, noun,	prod	yes	0.548

Tabella 3: Accuratezza media ottenuta per le 5 fold delle varie strategie adoperate

Il modello con la strategia appena descritta, ha ottenuto in fase di test, un'accuratezza di **0.65**. Anche con features basate su word embedding per l'allenamento, il modello SVM lineare è stato in grado di superare la baseline.

5 Fine-Tuning di un modello pre-addestrato

In questa sezione, si illustra l'ultimo punto del progetto, che consiste nel fare il fine-tuning di un language neural model di tipo Transformer. I modelli Transformer sono noti per il loro meccanismo di attenzione, che consente loro di concentrarsi sulle parti

rilevanti di un testo, migliorando la comprensione del contesto[Vaswani et al., 2017]. La libreria Transformer di Hugging Face offre una vasta selezione di modelli Transformer pre-addestrati. Durante la fase di fine-tuning, il modello viene ulteriormente addestrato per specializzarsi in un compito specifico. Affinché questa specializzazione avvenga, è necessario addestrare il modello in modo supervisionato utilizzando un dataset annotato.
Il fine-tuning, in questo caso, è servito ad adattare il modello al task della classificazione dell'ironia e il dataset annotato è quello fornito per lo shared Task di IronIta.

5.1 Workflow

Innanzitutto, è stato scelto il modello pre-addestrato della libreria Transformer da specializzare, ovvero Bert per l'italiano, addestrato su un corpus costituito da un recente dump di Wikipedia e vari testi dalla collezione di corpora OPUS[4].
Per lo sviluppo del codice relativo a questa sezione, è stato utilizzato un notebook di Google Colab per sfruttare la potenza di calcolo offerta dalle GPU.
In fase di pre-processing è stato passato il dataset al Tokenizer di Bert, per la tokenizzazione dei vari post. Una volta ottenuto il corpus tokenizzato, si è passati alla fase di training. Il modello è stato addestrato per 5 epoche. È stata anche condotta una piccola model selection in maniera manuale, per identificare i migliori valori di batch size, learning rate e weight decay. Finite le 5 epoche di training il modello è stato testato sul test set ufficiale di IronIta.

5.2 Risultati

Essendo i trasformer dei modelli molto sofisticati in grado di eseguire con successo la maggior parte dei task di NLP, le aspettative erano quelle di ottenere dei livelli di accuratezza abbastanza alti, almeno da superare i modelli precedenti basati su SVM lineari. Tuttavia, i risultati sono stati diversi dalle aspettative. La figura 2 riporta le

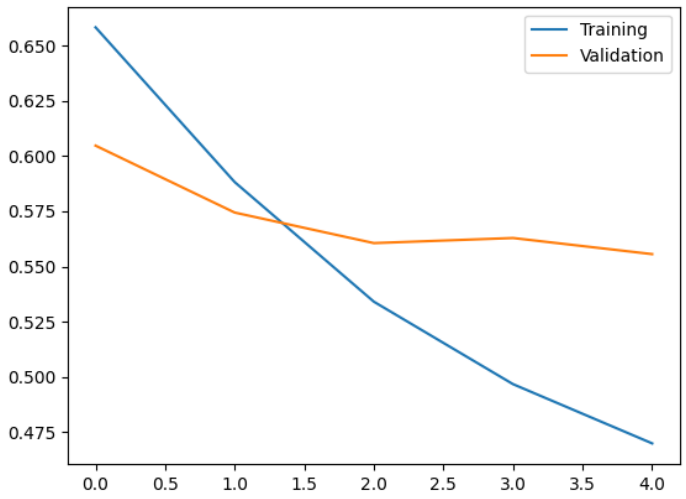

Figura 2: Curve di validation e training loss nel corso delle epoche di training

[4]https://huggingface.co/dbmdz/bert-base-italian-uncased

curve di training e di validation loss per la configurazione migliore trovata, ovvero una batch size di 32, un learning rate di 0.00001, e un weight decay di 0.2.
Dall'osservazione del grafico, si nota che la curva di loss mantiene un andamento abbastanza stabile fino alla terza epoca, dopo la quale inizia a salire leggermente. Questo suggerisce la possibilità di un caso di overfitting, e dunque potrebbe essere opportuno interrompere l'addestramento al termine della terza epoca.
Nella tabella 4 si riportano i valori di accuracy e f1-score per le 5 epoche. Anche per quanto riguarda le metriche di valutazione si nota come il punteggio di F1-score più alto sia stato raggiunto al momento della 3 epoca.

Epoch	**Accuracy**	**F1**
1	0.677	0.712
2	0.701	0.709
3	0.718	0.735
4	0.722	0.700
5	0.722	0.713

Tabella 4: Metriche di valutazione per ogni epoca di training

Per concludere, il modello fine-tuned ha raggiunto uno score di accuratezza di **0.68** in fase di testing. Dopo aver esplorato varie configurazioni di iperparametri e aver testato dimensioni del set di validazione diverse, i risultati poco soddisfacenti ottenuti da questo modello possono essere attribuiti alla complessità intrinseca del task stesso. L'ironia, come fenomeno linguistico, è spesso caratterizzata da un'ambiguità che può risultare difficile da cogliere anche per gli esseri umani, e questo sicuramente rappresenta una sfida significativa anche per i modelli neurali più avanzati disponibili attualmente

6 Conclusioni

Durante le varie fasi di questo progetto si sono esplorate diverse tecniche per lo sviluppo di modelli computazionali nell'ambito dell'elaborazione del linguaggio naturale. Gli SVM lineari hanno dimostrato di raggiungere risultati coerenti con le aspettative nel compito di classificazione dell'ironia. D'altra parte, il modello Transformer sottoposto a fine-tuning ha ottenuto prestazioni inferiori a quanto previsto, mettendo in evidenza la complessità intrinseca della classificazione dell'ironia. Questo task, infatti, si caratterizza per la presenza di forti elementi di ambiguità e soggettività.
Ad ogni modo il fine di questo progetto è dimostrare la comprensione e la messa in pratica delle metodologie di machine learning impiegate. Nonostante alcuni esperimenti abbiano mostrato risultati insoddisfacenti, possiamo considerare il progetto un successo in quanto sono state applicate le tecniche di programmazione apprese per lo sviluppo di modelli di NLP.

Riferimenti bibliografici

[Brunato et al., 2020] Brunato, D., Cimino, A., Dell'Orletta, F., Venturi, G., and Montemagni, S. (2020). Profiling-UD: a tool for linguistic profiling of texts. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 7145–7151. European Language Resources Association.

[Burges, 1998] Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167.

[Cignarella et al., 2018] Cignarella, A. T., Frenda, S., Basile, V., Bosco, C., Patti, V., and Rosso, P. (2018). Overview of the evalita 2018 task on irony detection in italian tweets (ironita). In *EVALITA@CLiC-it*.

[Comino et al., 2018] Comino, A., De Mattei, L., and Dell'Orletta, F. (2018). Multi-task learning in deep neural networks at evalita 2018. In *EVALITA Evaluation of NLP and Speech Tools for Italian: Proceedings of the Final Workshop 12-13 December 2018, Naples*.

[de Marneffe et al., 2014] de Marneffe, M.-C., Dozat, T., Silveira, N., Haverinen, K., Ginter, F., Nivre, J., and Manning, C. D. (2014). Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4585–4592. European Language Resources Association (ELRA).

[Lenci, 2018] Lenci, A. (2018). Distributional models of word meaning. *Annual Review of Linguistics*, 4(1):151–171.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.