Università degli Studi di Pisa

TEXT ANALYTICS

Professor G. Attardi, professor A. Esuli

AUTOMATIC MISOGYNY IDENTIFICATION

Daniele Gambetta
daniele.gambetta7@gmail.com 465137

Marta Marchiori Manerba
martamarchiori96@gmail.com 539412

A.A. 2019/2020

Indice

Introduzione

Automatic Misogyny Identification[1] (AMI) è stato proposto nell'ambito di **EVALITA 2018**[2], la competizione annuale promossa dall'*Associazione Italiana di Linguistica Computazionale* (AILC[3]) per la valutazione di strumenti NLP per la lingua italiana.
Il task si propone di identificare in modo automatico la presenza di contenuto misogino in tweet italiani e inglesi raccolti in due dataset distinti.
La competizione si articola in **due sotto-task:** 1) l'identificazione della **misoginia**, attraverso una classificazione binaria; 2) una classificazione multi-labels per individuare il **tipo** di misoginia e del **target** dell'offesa. La classe "target" può assumere due valori: "active" nel caso in cui l'utente destinatario sia specifico; "passive", nel caso in cui si tratti di un gruppo generico di persone. Per quanto concerne il tipo di comportamento misogino, riportiamo le cinque categorie indicate nelle linee guida del task[4]:

1. "*Stereotype & Objectification*: a widely held but fixed and oversimplified image or idea of a woman; description of women's physical appeal and/or comparisons to narrow standards.
2. *Dominance*: to assert the superiority of men over women to highlight gender inequality.
3. *Derailing*: to justify woman abuse, rejecting male responsibility; an attempt to disrupt the conversation in order to redirect women's conversations on something more comfortable for men.
4. *Sexual Harassment & Threats of Violence*: to describe actions as sexual advances, requests for sexual favours, harassment of a sexual nature; intent to physically assert power over women through threats of violence.
5. *Discredit*: slurring over women with no other larger intention."

La **relazione** inizia descrivendo i dataset a disposizione attraverso analisi linguistiche standard e grafici di vario tipo. Nella seconda sezione è presente il resoconto del preprocessing, attraverso vari tipi di tokenizzazioni e infine la costruzione degli embeddings attraverso BERT. Nella terza parte infine presentiamo i metodi di classificazione utilizzati (MLP, SVM, Decision Tree) e i risultati ottenuti. Il nostro lavoro verterà quindi nel **comparare la performance** raggiunta mediante i più moderni modelli computazionali **con i risultati "passati"**, ottenuti durante lo svolgimento della competizione[5].

[1] https://amievalita2018.wordpress.com/.
[2] http://www.evalita.it/2018.
[3] http://www.ai-lc.it/.
[4] https://amievalita2018.wordpress.com/do-you-ami/.
[5] Per i risultati ottenuti nel 2018: http://ceur-ws.org/Vol-2263/paper009.pdf.

1. Data Understanding

Data quality, distribuzione delle variabili e statistiche

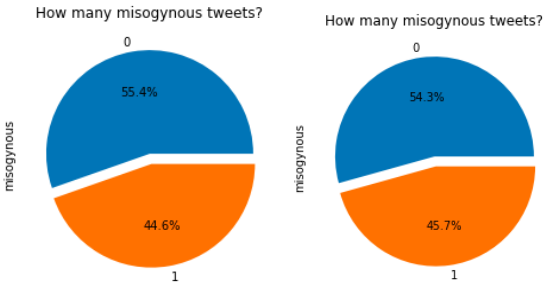
I **dati di training** forniti per la competizione comprendono **due dataset**: 4000 tweet in lingua italiana e 4000 tweet in lingua inglese, entrambi in formato **tsv**.
Ogni raccolta presenta **cinque campi**:

1. "id", di tipo int: l'identificatore numerico del tweet;
2. "text", di tipo object (string): la stringa contenente il testo del tweet;
3. "misogynous", di tipo int: identifica in modo binario (assumendo i valori 0 o 1) se il tweet è classificato come misogino;
4. "misogyny_category", di tipo object (string): presenta cinque categorie che identificano il tipo specifico di atteggiamento misogino presente nel tweet (questo valore di conseguenza è specificato soltanto per i tweet che presentano misogynous = 1: per i tweet non misogini ha valore di default uguale a 0);
5. "target", di tipo object (string): assume due possibili valori in base al destinatario dell'offesa, se specifico (active) o generico (passive). Inoltre, come per la precedente colonna, è specificato soltanto per i tweet misogini, mentre i tweet non misogini ha valore di default uguale a 0.

I dataset non presentano missing values.

Per descrivere la composizione e la distribuzione dei valori all'interno delle varie classi, abbiamo realizzato diversi **grafici** attraverso la libreria python Matplotlib[6].
Si noti in *Grafico 1* come la distribuzione della classe "misogynous" è quasi identica in entrambi i dataset. La distribuzione della classe "misogyny_category" in *Grafico 2* invece è fortemente sbilanciata in entrambe le raccolte, con differenze significative: nel dataset inglese il valore "discredit" prevale sugli altri, mentre nel dataset italiano i valori più frequenti sono "stereotype" e "discredit", seguiti da "sexual_harassment". La distribuzione della classe "target" in *Grafico 3* è differente: nei tweet italiani "active" è nettamente prevalente, mentre nei tweet inglesi la distribuzione è più bilanciata, nonostante "passive" sia comunque meno frequente. Per ulteriori indagini dei rapporti tra "target" e "misogyny_category", abbiamo realizzato delle crosstab (*Grafico 4*) e degli scatterplot (*Grafico 5*) per valutarne la correlazione.

Grafico 1 - Distribuzione della classe "misogynous" rispettivamente nella raccolta inglese e italiana:
Dataset inglese: 0 = 2215; 1 = 1785
Dataset italiano: 0 = 2172; 1 = 1828



[6] https://matplotlib.org/.

Grafico 2 - Distribuzione della classe "misogyny_category".

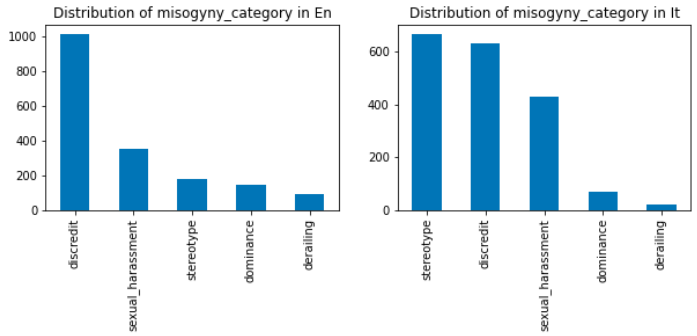


Tabella 1 - Distribuzione della classe "misogyny_category".

misogyny_category	**Dataset inglese**	**Dataset italiano**
discredit	1014	668
sexual_harassment	352	634
stereotype	179	431
dominance	148	71
derailing	92	24

Grafico 3 - Distribuzione della classe "target":
Dataset inglese: active = 1058; passive = 727
Dataset italiano: active = 1721; passive = 97

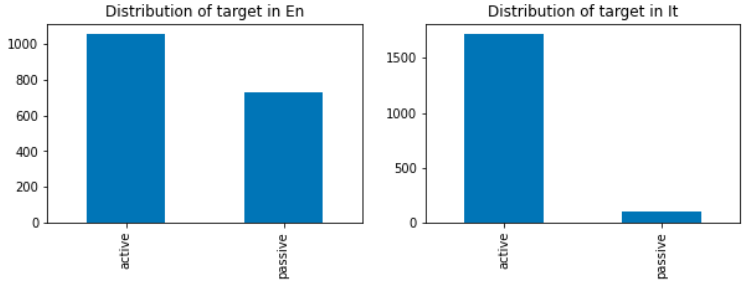


Grafico 4 - Distribuzione di "target" incrociata con "misogyny_category".

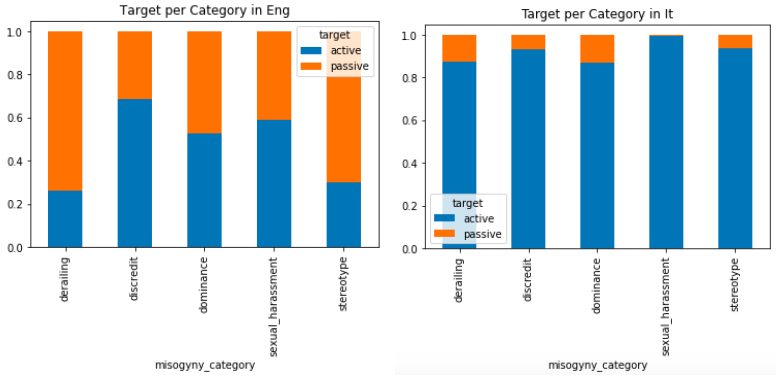
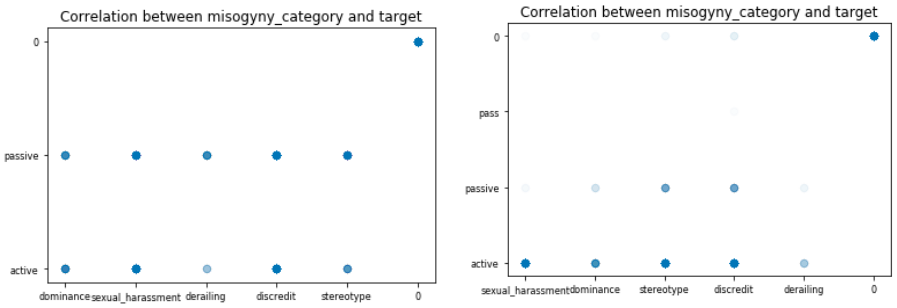


Grafico 5 - Scatterplot della classe "target" con "misogyny_category".



Analisi linguistiche

Per preparare il testo dei tweet ad analisi linguistiche specifiche, abbiamo utilizzato il modulo ***TweetTokenizer*** di **NLTK**[7], che applica un tokenizzatore concepito specificatamente per gestire la complessità linguistica nel "genere testuale" dei tweet (anche se per la lingua inglese, abbiamo scelto di applicarlo anche al dataset italiano, in mancanza di una versione language-specific).
Oltre al tokenizzatore, abbiamo applicato anche il Part of Speech Tagger e un algoritmo di stemming, per identificare eventuali informazioni di interesse[8].

Frequency Distribution

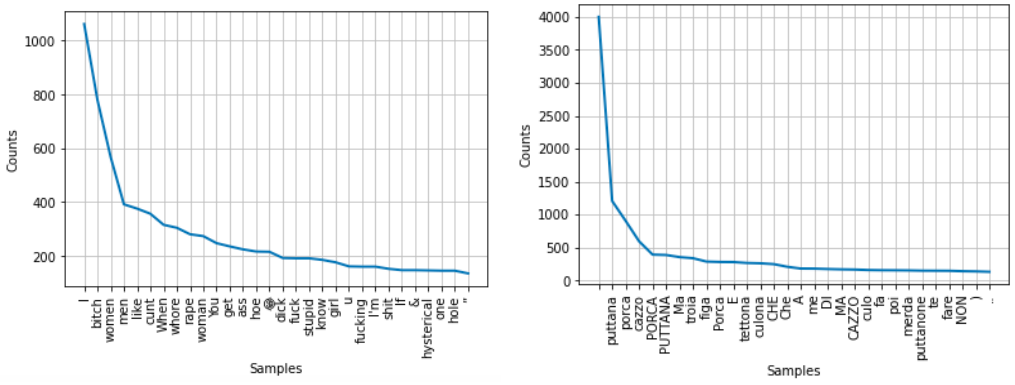
Attraverso la classe di NTLK ***FreqDist***[9], utilizzata per codificare le distribuzioni di frequenza, abbiamo individuato le parole tipo (il dizionario) e le relative occorrenze. Stampando i venti token più frequenti, abbiamo ottenuto per la maggior parte stop-words. Escludendole attraverso gli elenchi di stop-words standard forniti da NLTK[10], riportiamo di seguito i risultati. Si noti come le liste di NLTK per entrambe le lingue non siano esaurienti nell'escludere le parole "vuote": a titolo di esempio citiamo nel corpus italiano "Ma", "E", "CHE", "Che", "A", "DI".

Elenco 1 - Venti token più frequenti nel dataset inglese e italiano

```
[(' ', 4000),          [(' ', 4000),
 ('I', 1060),           ('puttana', 1207),
 ('bitch', 779),        ('porca', 906),
 ('women', 567),        ('cazzo', 595),
 ('men', 392),          ('PORCA', 394),
 ('like', 376),         ('PUTTANA', 388),
 ('cunt', 357),         ('Ma', 356),
 ('When', 316),         ('troia', 339),
 ('whore', 305),        ('figa', 288),
 ('rape', 281),         ('Porca', 282),
 ('woman', 274),        ('E', 281),
 ('You', 248),          ('tettona', 267),
 ('get', 236),          ('culona', 262),
 ('ass', 225),          ('CHE', 249),
 ('hoe', 217),          ('Che', 209),
 ('😂', 216),           ('A', 182),
 ('dick', 193),         ('me', 181),
 ('fuck', 192),         ('DI', 175),
 ('stupid', 192),       ('MA', 170),
 ('know', 186)]         ('CAZZO', 167)]
```

Le distribuzioni di frequenza dei 20 token listati nell'*Elenco 1* presentano il tipico andamento della legge di Zipf (nonostante il limitato numero di campioni): pochi termini ad altissima frequenza e la maggior parte dei termini presenti in frequenza nettamente minore.

Grafico 6 - Distribuzione di frequenza dei token riportati in Elenco 1



[7] http://www.nltk.org/api/nltk.tokenize.html.
[8] Rimandiamo al notebook per ulteriori approfondimenti.
[9] http://www.nltk.org/api/nltk.html.
[10] Da NLTK Data: http://www.nltk.org/nltk_data/.

Abbiamo scelto inoltre di utilizzare un'ulteriore intuitiva tecnica di visualizzazione, il wordcloud. Si noti l'alta frequenza di "https" in entrambi i dataset, che rivelano la presenza di URL in molti tweet. In evidenza inoltre forti termini offensivi, prevalentemente in riferimento alla sfera sessuale femminile.

Grafico 7 - Wordcloud dal dataset inglese e italiano.

Hashtag e retweet

Per evitare ripetizioni degli stessi **hashtag**, considerati differenti per l'uso delle maiuscole, riduciamo tutti gli hashtag in minuscolo per il dataset inglese che presenta numerose variazioni. Inoltre nel notebook abbiamo applicato una funzione[11] per separare le parole presenti negli hashtag, ma in fase di costruzione dell'embedding scegliamo di mantenerli in modo originale perché caratteristici di questo genere.

Si notino le significative differenze nei due dataset. Gli hashtag nei tweet inglesi fanno riferimento a noti movimenti femministi (#metoo) e sono fortemente collegati a riflessioni sul tema del maschilismo e della discriminazione di genere. Nel dataset italiano troviamo principalmente hashtag che richiamano reality show o programmi televisivi nazionali. Hashtag come #JuveReal, #Milan e #TorinoInter ricordano come questa collezione di tweet contenga anche testi dal contenuto neutro, non misogino e quindi in questo caso legato a eventi sportivi.

Elenco 2 - Hashtag più frequenti nel dataset inglese: 706.
Nel dizionario: 399
[('#womensuck', 123),
('#yesallmen', 29),
('#notallmen', 25),
('#metoo', 22),
('#maledominance', 19),
('#whore', 7),
('#rape', 7),
('#women', 6),
('#feminismiscancer', 6)]

Elenco 3 - Hashtag più frequenti nel dataset italiano: 792.
Nel dizionario: 422
[('#GF15', 38),
('#Amici17', 25),
('#isola', 16),
('#amici17', 13),
('#uominiedonne', 12),
('#JuveReal', 11),
('#chilhavisto', 9),
('#Milan', 9),
('#TorinoInter', 9)]

Si individuano i **retweet**, identificati dal prefisso "rt" o "RT". Tra gli utenti più frequentemente taggati nel dataset inglese troviamo Donald Trump e Hillary Clinton, entrambi personaggi politici; nel dataset italiano l'attrice Asia Argento e la deputata Laura Boldrini.

Elenco 4 - Retweet inglesi: 1964 occorrenze
[('@realDonaldTrump', 44),
('@Scouse_ma', 14),
('@HillaryClinton', 11),
('@FoxNews', 11),

Elenco 5 - Retweet italiani: 3611 occorrenze
[('@AsiaArgento', 92),
('@laraboldrini', 85),
('@Giuliettaxxx84', 46),
('@matteosalvinimi', 45),

[11] https://codereview.stackexchange.com/questions/119607/convert-hashtagstylephrases-to-a-proper-list-of-words.

('@themeredith', 10),
('@Yourfuckboy1', 9),
('@queensoverbitch', 9),
('@RepWilson', 9),
('@femfreq', 8)]

('@stanzaselvaggia', 39),
('@GILDA31395269', 32),
('@meb', 21),
('@1111scihl', 20),
('@tuttotuo63', 20)]

2. Preprocessing: tokenizzazione e utilizzo di BERT

Come descritto nell'*Introduzione*, abbiamo scelto di adottare la rappresentazione dei token tramite **embeddings** costruiti attraverso **BERT** (*Bidirectional Encoder Representations for Transformers*), l'encoder di Google[12], fornito da Transformers di *HuggingFace*[13]: abbiamo utilizzato la versione *BERT base-uncased*. Per l'italiano abbiamo utilizzato *BERT base-multilingual-uncased*[14], che identifica la lingua usata direttamente nel contesto testuale.

Abbiamo creato **tre** diversi **file**, ognuno contenente **un tipo diverso di preprocessing.** La **procedura per creare gli embeddings** attraverso BERT è la stessa: attraverso la funzione "encode" i token testuali vengono convertiti in identificatori numerici. Viene applicato il "padding", ovvero una volta individuato il testo più lungo, tutti i tweet nel corpus vengono allungati, attraverso l'aggiunta di zeri, per raggiungere questa massima lunghezza. In seguito viene applicata l'attention mask e viene computato l'embedding per tutto il corpus.
Si noti come ognuna dei preproccesing è stata eseguito sia per il dataset inglese che per quello italiano, per un totale di sei tokenizzazioni. Attraverso l'analisi delle performance dei modelli di classificazione, abbiamo infine scelto il preprocessing più rappresentativo ed efficace, in grado di ottenere i valori più alti.

BERT Tokenizer

La prima versione applica al testo raw il tokenizzatore di BERT, **BERT Tokenizer.** Nello specifico BERT utilizza come tokenizzatore *WordPiece*, una tecnica per segmentare le parole in sotto-livelli, per aiutare i modelli di NLP nel riconoscere la radice comune delle varie forme flesse di un unico lemma e possibilmente imparare dalla distribuzione delle forme più frequenti.

NLTK Tweet Tokenizer

Non essendo BERT Tokenizer specificatamente concepito per processare tweet, abbiamo applicato **Tweet Tokenizer**[15] di NLTK. Questo tokenizzatore applica automaticamente delle funzioni di normalizzazione: sostituisce sequenze di caratteri ripetuti con sequenze di lunghezza massima uguale a 3, rimuove gli username di Twitter dal testo, corregge l'eventuale presenza di codici HTML, etc.
Infine, utilizzando la funzione di BERT "convert_tokens_to_ids"[16], abbiamo trasformato i token testuali negli identificatori numerici e permesso così al modello di costruire l'embedding a partire dalla conversione numerica come input.

[12] https://arxiv.org/pdf/1810.04805.pdf.
[13] https://huggingface.co/transformers/.
[14] https://huggingface.co/transformers/multilingual.html.
[15] http://www.nltk.org/_modules/nltk/tokenize/casual.html#TweetTokenizer.
[16] https://huggingface.co/transformers/v2.3.0/main_classes/tokenizer.html#transformers.PreTrainedTokenizer.convert_tokens_to_ids.

Tokenizzazione custom

Per quanto riguarda la tokenizzazione custom, siamo partiti dal risultato dell'applicazione del Tweet Tokenizer di NLTK, aggiungendo **ulteriori modifiche**.
Abbiamo applicato il lowercase a tutto il corpus e abbiamo tolto i segni di punteggiatura e altri simboli: [!@$:).:,,?&]. Abbiamo inoltre rimosso la stringa "RT" o "rt" che identifica il fenomeno del retweet, tipico in Twitter, che consiste nel condividere o rispondere al contenuto di un altro utente. Infine abbiamo rimosso gli URL, attraverso le espressioni regolari, identificando i token che iniziavano per 'http'. Infine abbiamo rimosso le stop-words.
Come per il metodo precedente, abbiamo utilizzato la funzione "convert_tokens_to_ids" per ottenere la conversione numerica adatta come input al modello BERT.

Differenze nelle tokenizzazioni

A titolo informativo, riportiamo nei seguenti due screenshot il risultato delle tokenizzazioni, per evidenziarne le differenze. Tra le caratteristiche più evidenti, la presenza dei simboli "#" nella tokenizzazione effettuata da BERT, mentre nella tokenizzazione custom si distingue il trattamento delle maiuscole (attraverso l'applicazione del lowercase), la mancanza di stop-words e l'eliminazione dei simboli "@" che identificano le menzioni.

Tabella 2 - Output dei tre preprocessing per il dataset inglese.

	BERT Tokenizer	NLTK TweetTokenizer	TweetTokenizer + Costum
0	[please, tell, me, why, the, bitch, next, to, ...	[Please, tell, me, why, the, bitch, next, to, ...	[please, tell, me, why, the, bitch, next, to, ...
1	[@, emma, ##sha, ##rp, ##00, ##3, @, ld, ##rak...	[@emmasharp003, @Ldrake48Lee, Bitch, shut, the...	[emmasharp, 003, ldrake, 48lee, bitch, shut, t...
2	[@, ab, ##zd, ##af, ##ab, dear, cu, ##nt, ,, p...	[@abzdafab, Dear, cunt, ,, please, shut, the, ...	[abzdafab, dear, cunt, please, shut, the, fuck...
3	[rt, @, queen, ##of, ##dra, ##gon, ##sb, :, pl...	[RT, @queenofdragonsb, :, Pls, shut, the, fuck...	[queenofdragonsb, pls, shut, the, fuck, up, bi...
4	[rt, @, 21, ##bi, ##vc, ##k, :, ", when, u, go...	[RT, @21bIvck, :, ", when, u, gonna, get, your...	[21bivck, ", when, u, gonna, get, your, licens...
...	...	...	...
3995	[f, ##yi, ., maria, ##h, rhymes, with, par, ##...	[FYI, ., Mariah, rhymes, with, Pariah, which, ...	[fyi, mariah, rhymes, with, pariah, which, in,...
3996	[johnny, castle, loves, always, wet, warm, fuc...	[Johnny, Castle, loves, always, wet, warm, fuc...	[johnny, castle, loves, always, wet, warm, fuc...
3997	[@, don, ##can, ##non, @, dj, ##dra, ##ma, ole...	[@DonCannon, @DJDRAMA, Ole, fat, neck, ass, ni...	[doncannon, djdrama, ole, fat, neck, ass, nigh...
3998	[x, ##d, @, id, ##ub, ##bb, ##z, tight, hole, ...	[Xd, @Idubbbz, tight, hole, lost, <3, Kids, @m...	[xd, idubbbz, tight, hole, lost, <3, kids, met...
3999	[@, jon, ##ore, ##ad, @, dave, ##e, ##8, ##9, ...	[@jonoread, @Davee8989, Fat, cunt, most, likel...	[jonoread, davee, 8989, fat, cunt, most, likel...

Tabella 3 - Output dei tre preprocessing per il dataset italiano.

	BERT Tokenizer	NLTK TweetTokenizer	TweetTokenizer + Costum
0	[@, ka, ##sse, ##ma, ##min, ##4, @, lay, ##las...	[@KassemAmin4, @Laylasexgdr, Fatti, trovare, t...	[kassemamin, 4, laylasexgdr, fatti, trovare, t...
1	[@, me, ##b, tu, dovresti, ricominciare, dai, ...	[@meb, Tu, dovresti, ricominciare, dai, semafo...	[meb, dovresti, ricominciare, semafori, fare, ...
2	[amore, ,, sei, presenta, ##bile, ?, x, ##che,...	[Amore, ,, sei, presentabile, ?, Xchè, così, v...	[amoresei, presentabile, xchè, così, via, skyp...
3	[@, il, _, nulla, salvo, poi, mandare, la, cul...	[@Il_nulla, Salvo, poi, mandare, la, culona, a...	[il_nulla, salvo, poi, mandare, culona, mosca,...
4	[@, gior, ##gia, ##melo, ##ni, @, fratelli, ##...	[@GiorgiaMeloni, @FratelliditaIia, Vediamo, Ge...	[giorgiameloni, fratelliditaiia, vediamo, gent...
...	...	...	...
3995	[ho, ascoltato, tutto, il, giorno, cal, ##cut,...	[Ho, ascoltato, tutto, il, giorno, Calcutta, e...	[ascoltato, il, giorno, calcutta, mi, piaciuto...
3996	[@, mons, ##ta, x, ma, vi, sbriga, ##te, a, fa...	[@, monsta, x, ma, vi, sbrigate, a, fa, una, p...	[monsta, x, vi, sbrigate, fa, porca, puttana, ...
3997	[e, non, so, dove, studiare, perche, casa, mia...	[E, non, so, dove, studiare, perchè, casa, mia...	[non, so, studiare, perchè, casa, un, bordello...
3998	[la, cazzata, e, stata, davvero, grande, ., in...	[La, cazzata, è, stata, davvero, grande, ., In...	[cazzata, stata, davvero, grande, intendo, dim...
3999	[per, un, corso, serale, sono, in, classe, con...	[Per, un, corso, serale, sono, in, classe, con...	[un, corso, serale, in, classe, ragazzi, 25/28...

3 Classificazione

Rete Neurale

Abbiamo usato un classificatore *Multi-layer Perceptron* della libreria in *Python Scikit-Learn*[17], impostando una *GridSearch* con vari parametri (funzione di attivazione, numero di hidden layers e numero di nodi per layers, learning rate, ...).
Per ogni presetting abbiamo quindi ottenuto la configurazione migliore e completato la fase di addestramento del modello, usando una divisione del dataset in *train* e *split* con una proporzione 80/20.
Per ognuna delle tokenizzazioni effettuate tramite diversi preprocessing riportiamo quindi la tabella di valutazione.

Tabella 4 - Risultati task 1 (inglese e italiano)

Custom_en
Results on the train set:

	precision	recall	f1-score	support
0	0.75	0.71	0.73	1883
1	0.62	0.66	0.64	1317
accuracy			0.69	3200

Results on the test set:

	precision	recall	f1-score	support
0	0.70	0.70	0.70	432
1	0.65	0.65	0.65	368
accuracy			0.68	800

Custom_it
Results on the train set:

	precision	recall	f1-score	support
0	0.88	0.73	0.80	2097
1	0.61	0.81	0.70	1103
accuracy			0.76	3200

Results on the test set:

	precision	recall	f1-score	support
0	0.81	0.67	0.73	513
1	0.55	0.72	0.63	287
accuracy			0.69	800

Bert_eng
Results on the train set:

	precision	recall	f1-score	support
0	0.74	0.83	0.78	1583
1	0.81	0.71	0.76	1617
accuracy			0.77	3200

Results on the test set:

	precision	recall	f1-score	support
0	0.72	0.81	0.76	385
1	0.80	0.71	0.75	415
accuracy			0.76	800

Bert_it
Results on the train set:

	precision	recall	f1-score	support
0	0.78	0.80	0.79	1708
1	0.77	0.75	0.76	1492
accuracy			0.78	3200

Results on the test set:

	precision	recall	f1-score	support
0	0.80	0.79	0.79	428
1	0.76	0.77	0.76	372
accuracy			0.78	800

Nltk_en
Results on the train set:

	precision	recall	f1-score	support
0	0.89	0.67	0.77	2355
1	0.46	0.77	0.57	845

Nltk_it
Results on the train set:

	precision	recall	f1-score	support
0	0.79	0.78	0.79	1774
1	0.74	0.75	0.74	1426

[17] https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

accuracy 0.70 3200 Results on the test set: precision recall f1-score support 0 0.87 0.62 0.72 605 1 0.38 0.71 0.49 195 accuracy 0.64 800	accuracy 0.77 3200 Results on the test set: precision recall f1-score support 0 0.77 0.72 0.74 448 1 0.67 0.72 0.70 352 accuracy 0.72 800

Per quanto riguarda i sottotask, ci siamo concentrati su quello inglese, perchè avente dati più puliti e meglio distribuiti:

Tabella 5 - Risultati sottotask target (inglese)

Target
Results on the train set:
precision recall f1-score support

0 0.93 0.76 0.84 1046
1 0.56 0.84 0.67 382

accuracy 0.78 1428

Results on the test set:
precision recall f1-score support

0 0.90 0.70 0.79 263
1 0.48 0.79 0.60 94

accuracy 0.72 357

Fine Tuning

Abbiamo anche eseguito un algoritmo di *ktrain*[18] con *BERT-base-uncased*, che essendo molto pesante dato l'alto numero di parametri, dopo 3 epoche (2 ore) l'Accuracy non varia più, quindi, per evitare overfitting, ci siamo fermati ottenendo un buon risultato riportato in *Tabella 6.*

Tabella 6 - Fine tuning

```
Model: "tf_bert_for_sequence_classification"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bert (TFBertMainLayer)       multiple                  109482240
_________________________________________________________________
dropout_41 (Dropout)         multiple                  0
_________________________________________________________________
classifier (Dense)           multiple                  1538
=================================================================
Total params: 109,483,778
Trainable params: 109,483,778
Non-trainable params: 0
_________________________________________________________________
Train for 50 steps, validate for 25 steps
Epoch 1/10
50/50 [==============================] - 2217s 44s/step - loss: 0.5229 - accuracy: 0.7362 - val_loss: 0.4947 - val_accuracy: 0.7775
Epoch 2/10
50/50 [==============================] - 2199s 44s/step - loss: 0.3523 - accuracy: 0.8459 - val_loss: 0.4073 - val_accuracy: 0.8112
Epoch 3/10
50/50 [==============================] - 2202s 44s/step - loss: 0.1896 - accuracy: 0.9325 - val_loss: 0.4766 - val_accuracy: 0.8238
```

[18] https://pypi.org/project/ktrain

Bert come Classificatore

Abbiamo eseguito un'ulteriore prova di classificazione utilizzando direttamente BERT come classificatore, attraverso l'implementazione della libreria *SimpleTransformers*[19].
Di seguito i risultati ottenuti:

Tabella 7 - Risultati Bert come Classificatore

Inglese

	Classe predetta		
Classe reale	1	0	
1	269	101	370
0	77	353	430
	346	454	800

Precision	0.78
Recall	0.73
F1-score	0.75
Accuracy	0.78

Italiano

	Classe predettta		
Classe reale	1	0	
1	317	61	378
0	52	370	422
	369	431	800

Precision	0.86
Recall	0.84
F1-score	0.85
Accuracy	0.86

SVC

Infine abbiamo provato con un algoritmo *Support Vector Machine*[20], ottenendo risultati non molto migliori di MLP.

Tabella 8 - Risultati SVC

custom_en SVC

Results on the train set:

	precision	recall	f1-score	support
0	0.86	0.64	0.73	2385
1	0.40	0.69	0.50	815
accuracy			0.65	3200
3200				

Results on the test set:

	precision	recall	f1-score	support
0	0.84	0.61	0.71	586
1	0.39	0.67	0.49	214
accuracy			0.63	800

custom_it SVC

Results on the train set:

	precision	recall	f1-score	support
0	0.87	0.68	0.76	2265
1	0.49	0.77	0.60	935
accuracy			0.70	3200
200				

Results on the test set:

	precision	recall	f1-score	support
0	0.86	0.63	0.73	572
1	0.45	0.74	0.56	228
accuracy			0.67	800

[19] https://github.com/ThilinaRajapakse/simpletransformers
[20] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Decision Tree

Come criteri per il *Decision Tree*[21] abbiamo preso in considerazione il criterio Gini e il criterio Entropy.

Abbiamo condotto una **ricerca dei parametri ottimali** per ognuno dei due criteri, valutando le metriche per differenti valori di "min_samples_split" e "min_samples_leaf", in un range da 10 a 100, di dieci unità in dieci unità, attraverso un doppio loop (for split in range(10,100,10): for leaf in range(10,100,10):).

I parametri migliori **per il criterio Gini** sono: "min_samples_split"=20 e "min_samples_leaf"=10. La predizione sul training set, realizzata in 0.007 secondi, ha ottenuto per il dataset inglese i valori riportati nella *Tabella 8*. Per quanto riguarda la performance italiana, si registrano valori leggermente più alti, con Accuracy di 0.883.

Tabella 9 - Valutazione della performance del criterio Gini sul training inglese e confusion matrix.

```
Accuracy 0.8878125
F1-score [0.90080133 0.87090974]
              precision    recall  f1-score   support

           0       0.89      0.91      0.90      1785
           1       0.89      0.86      0.87      1415

    accuracy                           0.89      3200
   macro avg       0.89      0.88      0.89      3200
weighted avg       0.89      0.89      0.89      3200

array([[1630,  155],
       [ 204, 1211]])
```

La predizione sul test set avviene in 0.002 secondi. I risultati della performance sul dataset inglese sono riportati in *Tabella 9*. Per quanto riguarda l'italiano, si ottiene: Accuracy = 0.622.

Tabella 10 - Valutazione della performance del criterio Gini sul test inglese e confusion matrix.

```
Accuracy 0.585
F1-score [0.62780269 0.53107345]
              precision    recall  f1-score   support

           0       0.61      0.65      0.63       430
           1       0.56      0.51      0.53       370

    accuracy                           0.58       800
   macro avg       0.58      0.58      0.58       800
weighted avg       0.58      0.58      0.58       800

array([[280, 150],
       [182, 188]])
```

Per il criterio Entropy, i parametri ottimali individuati sono: "min_samples_split"=10 e "min_samples_leaf"=10. La predizione sul training è realizzata in 0.006 secondi. I risultati per il dataset inglese sono riportati in *Tabella 10*. Per l'italiano, si registra: Accuracy = 0.909.

[21] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Tabella 11 - Valutazione della performance del criterio Entropy sul training inglese e confusion matrix.

```
Accuracy 0.8971875
F1-score [0.91023192 0.8797075 ]
              precision    recall  f1-score   support

           0       0.89      0.93      0.91      1785
           1       0.91      0.85      0.88      1415

    accuracy                           0.90      3200
   macro avg       0.90      0.89      0.89      3200
weighted avg       0.90      0.90      0.90      3200

array([[1668,  117],
       [ 212, 1203]])
```

La predizione sul test avviene in 0.002 secondi. I risultati per l'inglese in *Tabella 11*, mentre per l'italiano: Accuracy = 0.607.

Tabella 12 - Valutazione della performance del criterio Entropy sul test inglese e confusion matrix.

```
Accuracy 0.5975
F1-score [0.64537445 0.53468208]
              precision    recall  f1-score   support

           0       0.61      0.68      0.65       430
           1       0.57      0.50      0.53       370

    accuracy                           0.60       800
   macro avg       0.59      0.59      0.59       800
weighted avg       0.60      0.60      0.59       800

array([[293, 137],
       [185, 185]])
```

Conclusioni

Nell'approcciarsi a questa competizione, abbiamo inizialmente analizzato i dati, sperimentando e costruendo un nostro modello a partire dalle tre tokenizzazioni create: l'intenzione è stata quella di **indagare il ruolo e l'impatto che scelte di processing hanno sulla performance finale**.
La **nostra performance** migliore sulla valutazione del test ha raggiunto con **MLP** per l'inglese 0.76, mentre per l'italiano 0.78, entrambi con la tokenizzazione **BERT**.
Utilizzando invece **BERT come classificatore**, abbiamo ottenuto per l'inglese un'Accuracy di 0.78, leggermente più alta della precedente, mentre per l'italiano 0.86, registrando un significativo miglioramento nella performance.

Per completezza, possiamo osservare i risultati[22] riportati alla fine della **competizione nel 2018**. Per il **sottotask in lingua inglese,** il miglior risultato è stato ottenuto dal team *hateminers*, con un'Accuracy del 0.70; **nell'italiano** invece dal team *bakarov* con un'Accuracy del 0.84. Cimentarsi in questo task è stato impegnativo ma anche soddisfacente e per questo abbiamo intenzione di provare a partecipare alla nuova edizione, **AMI 2020**[23].

[22] https://amievalita2018.wordpress.com/evaluation-and-results/.
[23] https://amievalita2020.github.io/#about.

Riferimenti

Bibliografia

- **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**, *Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova - Google AI Language.* 2019 (https://arxiv.org/pdf/1810.04805.pdf)
- **The Problem of Identifying Misogynist Language on Twitter (and other online social spaces)**, *Sarah Hewitt, Dr T Tiropanis, Dr C Bokhove*. 2016 (https://dl.acm.org/doi/pdf/10.1145/2908131.2908183)

Sitografia

- https://github.com/google-research/bert
- https://github.com/kpot/keras-transformer
- https://huggingface.co/transformers/
- https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/
- https://github.com/jalammar/jalammar.github.io/blob/master/notebooks/bert/A_Visual_Notebook_to_Using_BERT_for_the_First_Time.ipynb
- https://jalammar.github.io/illustrated-bert/
- https://jalammar.github.io/illustrated-transformer/
- https://openai.com/blog/language-unsupervised/
- https://stackabuse.com/text-classification-with-bert-tokenizer-and-tf-2-0-in-python
- https://datascienceplus.com/twitter-analysis-with-python/
- http://blog.chapagain.com.np/python-nltk-twitter-sentiment-analysis-natural-language-processing-nlp/