Università degli Studi di Milano - Bicocca
Scuola di Scienze
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di laurea in Informatica

Enhancing irony detection with affective information

Relatore: Prof. Elisabetta Fersini

Relazione della prova finale di:
Gianluca Giudice
Matricola 830694

Anno Accademico 2019-2020

Contents

Chapter 1

Introduzione

1.1 Descrizione del problema

La sentiment analysis è un campo dell'elaborazione del linguaggio naturale (NLP) che si occupa di costruire sistemi per l'analisi di un testo, ha il fine di identificare e classificare l'informazione come il sentimento e l'opinione espressa nello stesso. Si basa sui principali metodi di linguistica computazionale e di analisi testuale. L'analisi del sentiment è utilizzata in molteplici settori: dalla politica ai mercati azionari, dal marketing alla comunicazione, dall'analisi dei social media alla valutazione delle preferenze del consumatore.

Il riconoscimento automatico dell'ironia nei contenuti generati da utenti, è uno dei compiti più complessi per quanto riguarda l'elaborazione del linguaggio naturale. Tuttavia è di fondamentale importanza per tutti i sistemi di sentiment analysis, in quanto facendo uso dell'ironia è possibile invertire completamente la polarità di una propria opinione, facendola passare da positiva a negativa e viceversa. Diventa pertanto cruciale sviluppare dei sistemi di sentiment analysis che sono consapevoli del fenomeno e in grado di riconoscerlo.

L'ironia è un tema studiato in diverse discipline, come la linguistica, filosofia e psicologia, ma è difficile da definire formalmente, soprattutto per questo motivo ne è difficile il riconoscimento. Nonostante ciò ci sono basi teoriche che suggeriscono il ruolo importante della sfera emozionale nell'uso dell'ironia, quindi un fattore chiave per riconoscerlo. Con questo si intende anche un uso indiretto e non esplicito del carico emotivo in ciò che si vuole comunicare.

I social netowork in generale, e twitter nello specifico, sono ampiamente utilizzati come fonte di informazione per comprendere la web reputation di un brand, perciò si prestano bene per la sperimentazione di modelli computazionali per il riconoscimento dell'ironia, essendo di fatti una grande risorsa per quanto riguarda i dati testuali generati da utenti.

1.2 Approccio al problema

Si può considerare il riconoscimento dell'ironia come un problema di classificazione. Una frase potrà quindi essere classificata come appartenente alla classe ironica o non ironica. Si noti che per come viene approcciato il problema, l'ironia è considerata al pari del sarcasmo.

A questo scopo, la classificazione avviene con l'utilizzo di tecniche che fanno parte dell'intelligenza artificiale. Nello specifico, vengono creati dei modelli supervisionati di machine learning. Tramite l'utilizzo di algoritmi, vengono utilizzate grande quantità di dati così da estrarre dei pattern presenti in quest'ultimi.

Come già detto, twitter è una risorsa che fornisce moltissimi contenuti generati da utenti. Viene sfruttato questo aspetto creando i modelli matematici in grado di far apprendere alla macchina dai dati, per fa sì che sia possibile riconoscere l'ironia nel testo.

Il codice scritto in python, utilizzato per l'estrazione delle features, il training dei modelli e la creazione della matrice su cui è stata fatta PCA, oltre ai notebooks utilizzati per l'analisi dei report e il risultato della PCA, è disponibile in questo repository[1] su github.

1.3 Sintesi dei risultati

Il risultato degli esperimenti mostra come per il riconoscimento dell'ironia si ottengo i risultati migliori combinando tutte le features considerate. Inoltre le performance variano decisamente a seconda dell'algoritmo utilizzato per creare i classificatori. Nel caso preso in considerazione, l'algoritmo Support Vector Machine con un kernel lineare risulta essere il migliore tra quelli analizzati.

[1] Irony detection repository: `www.github.com/gianlucagiudice/irony-detection`

Chapter 2

Stato dell'arte

2.1 Approccio supervisionato

2.2 Approccio semi supervisionato

2.3 Approccio non supervisionato

Chapter 3

Sistema realizzato

3.1 Descrizione del sistema proposto

Per riconoscere l'ironia nel testo, vengono creati dei modelli supervisionati di machine learning. Partendo dai dati a disposizione si utilizzano delle tecniche di preprocessing che consistono nel pulire il testo di partenza e prepararlo per essere dato in input ai vari classificatori. Inoltre si estraggono delle features da ogni messaggio, le quali più sono discriminanti nel riconoscimento dell'ironia, meglio distinguono la classe di appartenenza.

Il dataset viene diviso in due parti: training set e test set. La prima ha dimensione decisamente maggiore e viene utilizzata per far apprendere uno specifico classificate, l'altra per testarlo. Questo è importante in quanto un determinato modello dovrà riconoscere l'ironia in un generico documento e non solo tra quelli a disposizione. Si noti che entrambe le porzioni sono composte sia dal testo già codificato e processato che le features estratte, hanno solo scopi diversi quando si tratta di creare il classificatore.

Vengono quindi creati vari modelli supervisionati di machine learning fornendo in input il dataset di training. Avendo costruito più classificatori che utilizzano diversi algoritmi per apprendere dai dati, è possibile confrontarne le performance per valutare il migliore tra tutti, ovvero quello che meglio distingue tra documenti ironici e non ironici.

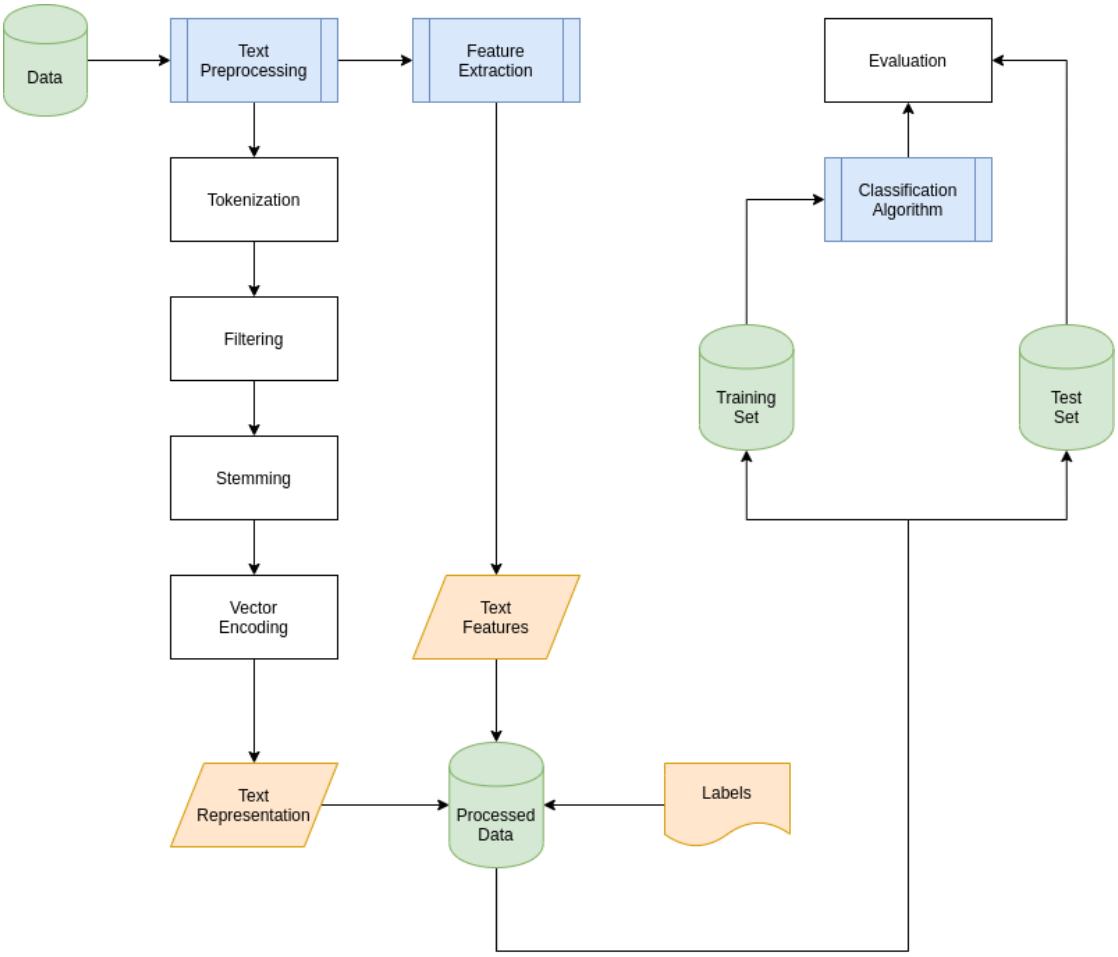


Figure 3.1: Workflow.

3.2 Dati in input

Per procedere con la costruzione di modelli supervisionati per il riconoscimento dell'ironia, è necessaria una collezione di testi (in questo caso tweet) che insieme alle relative etichette associate ad ogni documento, costituisce il dataset. Questo aspetto è cruciale in quanto, una volta essere stati opportunamente processati, saranno proprio questi i dati forniti in input al classificatore, e hanno lo scopo di farlo apprendere e quindi successivamente riconoscere l'ironia in testi che non ha mai visto prima.

Tweet	**Label assocaita**
Lorem ipsum dolor sit amet, consectetur adipiscing elit	ironico
Cras ornare turpis ut odio finibus, viverra porta felis lobortis	ironico
Suspendisse in ex a felis porta convallis	non ironico
Nam laoreet, lacus at ullamcorper iaculis, id interdum nisl elit nec elit.	non ironico

Table 3.1: Esempio di dataset a disposizione.

3.3 Rappresentazione del testo

Il corpora a disposizione è definito "crudo" e non può essere direttamente utilizzato la fase di training. Prima è necessario codificare il testo per ottenere una rappresentazione numerica da dare in input ad un algoritmo di classificazione per la fase di training e test. A questo scopo sono state utilizzate due diverse tecniche

3.3.1 Rappresentazione Bag-of-word

La prima tecnica di rappresentazione utilizzata è boolean bag-of-words. Il testo viene codificato come una matrice booleana costituita dai documenti sulle righe e i tokens sulle colonne. I tokens sono tutte quelle parole appartenenti ad un dizionario costruito a partire dal corpora, tutte le operazioni eseguite sono spiegate di seguito.

3.3.1.1 Tokenization

La tokenizzazione è un passo preliminare per l'elaborazione computazionale del testo. Tokenizzare vuol dire dividere le sequenze di caratteri in unità minime di analisi dette "token". Un approccio potrebbe essere considerare un token come una sequenza di caratteri delimitata da spazi, tuttavia una tale definizione lascia spazio a numerose eccezzione, come ad esempio la presenza di punteggiatura. Per questa operaziona viene pertanto utilizzata la libreria nltkl[1] in grado di gestire correttamente tutti questi casi limite, oltre a poter sfruttare una Twitter-aware tokenization adattandosi quindi bene al dominio in questione.

[1]Natural Language Toolkit: `www.nltk.org`

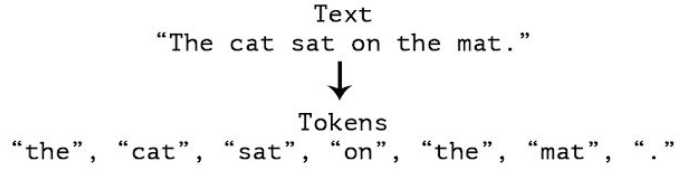


Figure 3.2: Text tokenization.

I vari tokens estratti vengono convertiti in lowercase e sono considerati validi solo se in match con il pattern regex `[a-z]+`. Così facendo non si considerano gli hashtag, la punteggiatura, le emoji e tutto ciò che non è considerato una parola. Si noti che potrebbero verificarsi casi in cui lo stesso token è presente in documenti diversi, questo è ragionevole in quanto le stesse parole possono essere presenti in tweet diversi.

Il risultato ottenuto è una lista di tokens validi contenuti in ogni tweet:

Tweet	**Tokens lowercase**
Lorem ipsum dolor sit amet, consectetur adipiscing elit	$[t_1, t_2, t_3, t_4, t_5]$
Cras ornare turpis ut odio finibus, viverra porta felis lobortis	$[t_6, t_7, t_1, t_8, t_9, t_4]$
Suspendisse in ex a felis porta convallis	$[t_{10}, t_6, t_{11}, t_{12}]$
Nam laoreet, lacus at ullamcorper iaculis.	$[t_{13}, t_{14}, t_8, t_{15}]$

Table 3.2: Esempio di tokens associati ad ogni tweet.

Da qui si crea l'insieme di tokens univoci.

3.3.1.2 Filtering

L'insieme di termini univoci ottenuto è filtrato in base ai seguenti criteri:

- **Occorrenza minimia** fissato valore di threshold

  Viene fissato un valore di soglia (Es. 10) e si scartano tutte quelle parole che compaiono tra tutti i testi meno del valore scelto.

- **Stopwords + RT**

  Vengono scartate tutte le parole più comuni che sono generalmente presenti in una frase. Queste sono un insieme predefinito contenente parole come gli articoli, proposizioni, pronomi e verbi ausiliari.

  Es: $\{a,\ the,\ of,\ is,\ into,\ it,\ ...\}$

  Dato il dominio preso in considerazione, è possibile che un utente compia l'operazione di ReTweet. Questo è codificato nei messaggi dall'abbreviazione *"rt"*. Non ritenendolo un aspetto rilevante per il riconoscimento dell'ironia, viene anch'essa considerata una stopwords, e di conseguenza non trattato come token valido.

- **Irony** e **Ironic**

  Essendo il dataset a disposizione etichettato mediante la tecnica self-tagging (section 5.1), è presente l'hashtag *#irony* in tutti i tweet ironici. Pertanto nell'insieme dei tokens validi verranno escluse le parole *irony* e *ironic*, così da non avere un bias nei dati sotto questo aspetto. Nel caso contrario i dati usati per creare i modelli avrebbero a disposizione una

componente che ben distingue i tweet ironici, ma che nella realtà non sarebbe presente, in quanto non è decisamente corretto assumere che ogni testo ironico contenga una delle due parole.

3.3.1.3 Stemming

Lo stemming è il processo di riduzione della forma flessa di una parola alla sua forma radice.

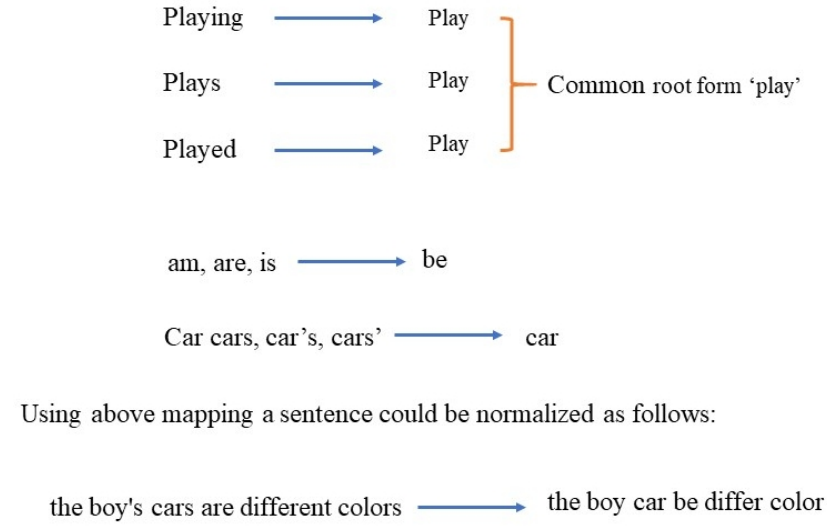


Figure 3.3: Words stemming. [2]

Con questa tecnica è possibile ridurre il numero di tokens, dal momento che le stesse parole con suffisso diverso verranno mappate nella medesima radice. Inoltre non è necessario che una parola dopo il processo di stemming venga trasformata in un'altra di senso compiuto, quello che conta è la semantica del token.

3.3.1.4 Vector encoding

L'insieme delle parole ottenute costituisce il dizionario del corpora. Si procede costruendo una matrice booleana composta dai tweets sulle righe e le parole del dizionario sulle colonne.

Sia $D = \{w_1,\ w_2, ...,\ w_m\}$ il dizionario individuato dall'insieme delle n parole univoche.
Sia $T_i = \{t_{i0}\ , t_{i1},\ ...,\ t_{ik}\},\ 1 \leq i \leq n$, l'i-esimo tweet composto da k tokens opportunamente stemmatizzati.
Viene definita la matrice $M_{n,m}$

$$M[i,j] = \begin{cases} 1 & \text{se } w_j \in T_i \\ 0 & \text{altrimenti} \end{cases} \quad t.c\ 1 \leq i \leq n,\ 1 \leq j \leq m$$

[2] Fonte: `www.datacamp.com/community/tutorials/stemming-lemmatization-python`.

3.3.2 Rappresentazione mediante transformer

3.3.2.1 BERT

BERT (Bidirectional Encoder Representations from Transformers) è un modello di deep learning creato da Google. A partire da un documento, una volta generati i tokens, è possibile fornire questi come input al modello per generare word embeddings, ovvero un vettore di numeri reali associato ad ogni token. Questo modello viene sfruttato per utilizzare gli embeddings come features, sostituendoli alla rappresentazione del testo BOW.

La rete neurale è composta da 12 hidden layers, ciascuno di questi fornisce in output un vettore di 768 elementi. Sorge quindi il problema della scelta del vettore da utilizzare, inoltre per gli emebddings vengono generati per ciascuno dei tokens. Bisogna pertanto adottare la corretta pooling strategy:

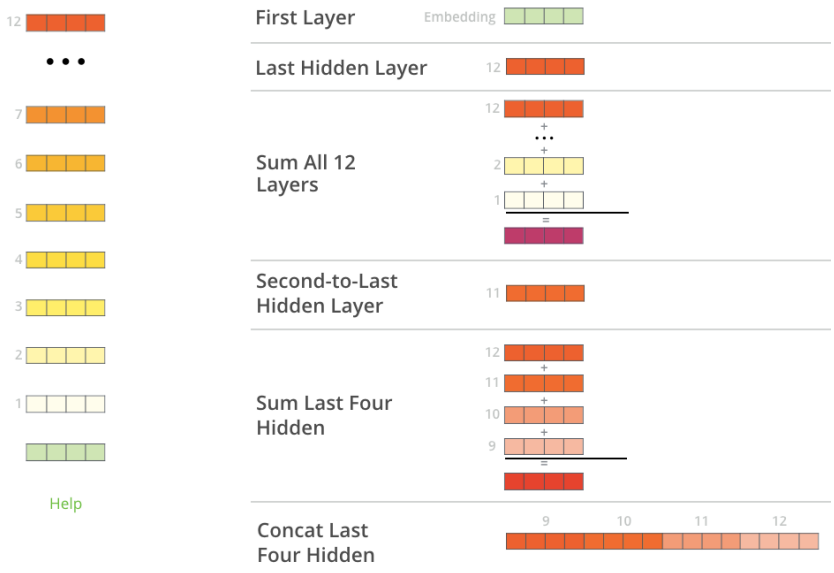


Figure 3.4: Pooling strategy. [3]

Nel caso in questione, si è scelto di concatenare gli ultimi 4 hidden layers e mediare tutti gli embeddings dei tokens presenti nel documento. In tutto si ottiene un vettore di 3072 elementi, il quale sarà utilizzato come features dei modelli in sostituzione della rappresentazione BOW.

3.3.2.2 Sentence-BERT

Sentence-BERT[4] è una modifica della rete neurale BERT. Questa messa a punto del modello, permette di generare embeddings semanticamente validi relativi ad intere frasi.

3.4 Caratteristiche linguistiche

La scelta di features ad alto contenuto informativo ed indipendenti tra loro, è un passo fondamentale per un corretto riconoscimento di pattern nei dati. A questo scopo vengono estratte

[3] Fonte: `www.jalammar.github.io/illustrated-bert`.
[4] Sentence-BERT: `www.arxiv.org/abs/1908.10084`

delle caratteristiche linguistiche dal testo che sono rilevanti per il riconoscimento dell'ironia. Più le caratteristiche sono discriminanti quando si esprime un messaggio ironico, meglio si potranno distinguere le due classi. Queste features, insieme alla codifica testuale, verranno usate per creare il modello.

3.4.1 PP (Pragmatic particles)

Per catturare il senso non letterale racchiuso in ogni messaggio, vengono considerati alcuni elementi linguistici di seguito elencati.

3.4.1.1 Emoticons

Le emoticons sono riproduzioni stilizzate di quelle principali espressioni facciali umane che esprimono un'emozione. In accordo con l'assunzione che esista una relazione tra il sentimento espresso e l'ironia, queste vengono distinte in due categorie:

- Positive

  E.g.: ':-)', ':-D'

- Negative

  E.g.: ':-(', ':=('

3.4.1.2 Acronimi

Gli acronimi sono un ulteriore strumento della comunicazione non verbale. Come le emoticons vengono presi in considerazione dal momento che possono esprire sentimenti positivi o negativi, infatti possiamo anch'essi dividerli in due categoire:

- Positivi

  E.g.: 'ROLF' (Rolling On Floor Laughing), 'LHO' (Laughing Head Off)

- Negativi

  E.g: 'BM' (Bad Manner)

3.4.1.3 Espressioni onomatopeiche

L'uso di espressioni onomatopeiche come 'boom' o 'clap' possono essere utilizzate nell'esprimere un messaggio ironico. Tuttavia in questo caso viene presa in considerazione solo la frequenza di queste espressioni, al contrario di come visto in precedenza per le emoticons e gli acronimi, dove si valutava anche la polarità.

3.4.1.4 Punteggiatura

La punteggiatura nei social media non segue le convenzioni ortografiche:
E.g.: 'Do you hear us now ?????'
Per questa ragione viene contata la frequenza di virgole, punti di domanda e punti esclamativi.

3.4.1.5 Estrazione particelle pragmatiche

Di seguite viene mostrato un esempio di come le particelle pragmatiche vengono estratte da un messaggio:

```
Tweet: ':D lmao! RT @KarlDetkenProDJ iPad Humor Steve, I'ma let you finish,
       but Moses had the greatest  tablet of all time ;) clap clap'
Emot (−, +)  >>> [0, 1]
Init (−, +)  >>> [0, 1]
Onom (#)     >>> [2]
Punct        >>> {',': 2, '!': 1, '?': 0}
```

Listing 3.1: Esempio di tweet processato per estrarre le particelle pragmatiche.

3.4.2 POS (Part Of Speech)

La struttura delle frasi che costituiscono il tweet, può essere un indicatore per il riconoscimento dell'ironia. A questo scopo viene tenuta in considerazione la frequenza dei POS (part-of-speech) tag relativi ai tokens parole presenti in un messaggi.

Per associare i POS tag ad ogni tweet viene utilizzato un POS tagger[5] supervisionato specifico per il dominio considerato. Si procede con una tokenizzazione del tweet per poi associare ad ognuno di questi il corretto tag. Di seguito la lista di tags considerati:

Nominale

- **N**: Nome comune
- **O**: Pronome personale
- **^**: Nome proprio
- **S**: Nominale + Possessiva
- **Z**: Nome proprio + Possessiva

Closed-class words

- **D**: Determiner
- **P**: Pre/Post-posizione; Congiunzione subordinante
- **&**: Congiunzione coordinante
- **T**: Verb particles
- **X**: Existential there

Open-class words

- **V**: Verbo
- **A**: Aggettivo
- **R**: Avverbio
- **!**: Interjection

Composti

- **L**: Nominale + Verbo (e.g.: I'm)
- **M**: Nome proprio + Verbo
- **Y**: X + Verbo

Di seguito è mostrato un esempio di POS tags associati al tweet passando per i tokens:

```
Tweet: 'Need a second opinion? Who gave you the first one?'
Tokens  >>> [Need], [a], [second], [opinion], [?], [Who],
            [gave], [you], [the], [first], [one], [?]
```

[5] ARK Twitter Part-of-Speech Tagger: www.ark.cs.cmu.edu/TweetNLP/

```
Tags    >>> ['V', 'D', 'A', 'N', ',', 'O', 'V', 'O', 'D', 'A', '$', ',']
Freq.   >>> {'N': 1, 'O': 2, '^': 0, 'S': 0, 'Z': 0, 'V': 2,
             'A': 2, 'R': 0, '!': 0, 'D': 2, 'P': 0, '&': 0,
             'T': 0, 'X': 0, 'L': 0, 'M': 0, 'Y': 0}
```

Listing 3.2: Esempio di tweet con POS tag associte.

Come si può notare, il POS tagger associa dei tag che non sono stati elencati sopra, i verranno semplicemente ignorati e non utilizzati come features per il modello.

3.4.3 EMOT (Emotional Features)

Alcune teorie psicologiche suggeriscono l'esistenza di emozioni primarie. Per questo motivo sono stati utilizzati i lessici EmoLex e EmoSenticNet, in cui vengono riportate le sfere emotive a cui appartengono le parole. I lessici si possono intendere come diversi insiemi che caratterizzano le emozioni primarie, dove ognuno di questi insiemi è composto dalle parole che sono considerate appartenenti a quella specifica categoria emozionale. Inoltre gli insiemi considerati non devono necessariamente essere disgiunti.
Qualche esempio di seguito:

$birthday \in JOY \wedge birthday \in SURPRISE \wedge birthday \in ANTICIPATION$
$honor \in TRUST$
$stair \notin (JOY \cup SURPRISE \cup ANTICIPATION \cup TRUST)$

Come feature aggiuntiva per il riconoscimento dell'ironia, dato un tweet si considerano tutti i tokens validi che rappresentano le parole. Quindi si procede consultando i lessici per ognuna delle parole, così da recuperate le sfere emotive a cui è essa associata. Una volta eseguita l'operazione per tutte queste, si calcola la frequenza totale di ogni emozione all'interno del tweet.

Le due risorse si basano su differenti teorie che identificano ognuna le proprie emozioni primarie, per questo motivo si combina il risultato di entrambe sommando le frequenze ottenute da ognuna delle due risorse.

Emozione	**Lessico**
Anger	EmoLex, EmoSenticNet
Anticipation	EmoLex
Disgust	EmoLex, EmoSenticNet
Fear	EmoLex, EmoSenticNet
Joy	EmoLex, EmoSenticNet
Sadness	EmoLex, EmoSenticNet
Suprise	EmoLex, EmoSenticNet
Trust	EmoLex

Table 3.3: Emozioni primarie dei lessici.

Un esempio di come vengono estratte le features emozionali da un tweet:

```
Tweet: 'I nominate @MrsStephenFry for a Shorty Award in because... why
       not? http://bit.ly/shorty'
Words         >>> ['nominate', 'shorty', 'award']
EmoLex        >>> {'anger': 0, 'anticipation': 1, 'disgust': 0, 'fear': 0,
                   'joy': 1, 'sadness': 0, 'surprise': 1, 'trust': 1}
EmoSenticNet >>> {'anger': 0, 'disgust': 0, 'joy': 3, 'sadness': 0,
```

```
        'surprise': 0, 'fear': 0}
Combined    >>> {'sadness': 0, 'anticipation': 1, 'fear': 0, 'anger': 0,
        'joy': 4, 'disgust': 0, 'trust': 1, 'surprise': 1}
```

Listing 3.3: Esempio di estrazione delle emozioni da un tweet.

Chapter 4

Modelli supervisionati

Il machine learning è lo studio di algoritmi che migliorano automaticamente attraverso l'esperienza, ed è sottoinsieme dell'intelligenza artificiale. Il machine learning crea modelli matematici basati sui dati, i quali vengono utilizzati per far apprendere alla macchina, con lo scopo di fare previsioni o decidere riguardo dati che non sono mai stati visti prima.

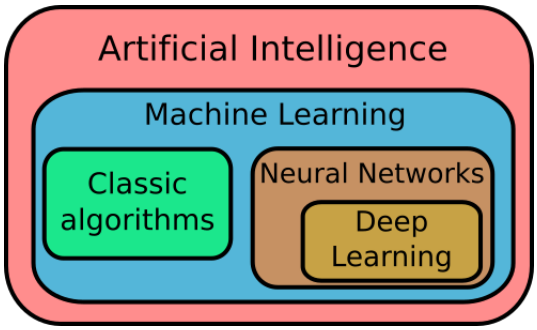


Figure 4.1: Diagramma intelligenza artificiale.[1]

Nello specifico i modelli supervisionati fanno parte di quell'insieme di algoritmi "classici" che per apprendere dai dati ed estrarre dei pattern da questi, oltre ai dati di training in input, hanno bisogno delle relative etichette. Questa informazione aggiuntiva viene usata nella fase di apprendimento e indica a priori la relativa classe di appartenenza delle istanze che compongono il dataset. Nel caso dell'ironia si fa riferimento ad algoritmi supervisionati di machine learning per risolvere un problema di classificazione, ovvero dato un tweet classificarlo come "ironico" o "non ironico".

Esistono diversi algoritmi per creare modelli supervisionati, quelli che sono stati utilizzati negli esperimenti sono:

- Decision Tree
- Multinomial Naive Bayes
- Bayesian Network
- Support Vector Machine

[1] Fonte: `www.mc.ai/machine-learning-in-5-minutes/`.

4.1 Decision Tree

Un albero di decisione è un modello predittivo con una struttura ad albero. Ogni nodo interno rappresenta una feature, un arco verso un nodo figlio rappresenta un possibile valore per quella proprietà, ovvero una decisione, e una foglia il valore predetto. Il processo consiste in una sequenza di test, comincia sempre dal nodo radice, e procede verso il basso. A seconda dei valori rilevati in ciascun nodo, si otterrà un path che porterà ad un nodo foglia e quindi una classificazione.

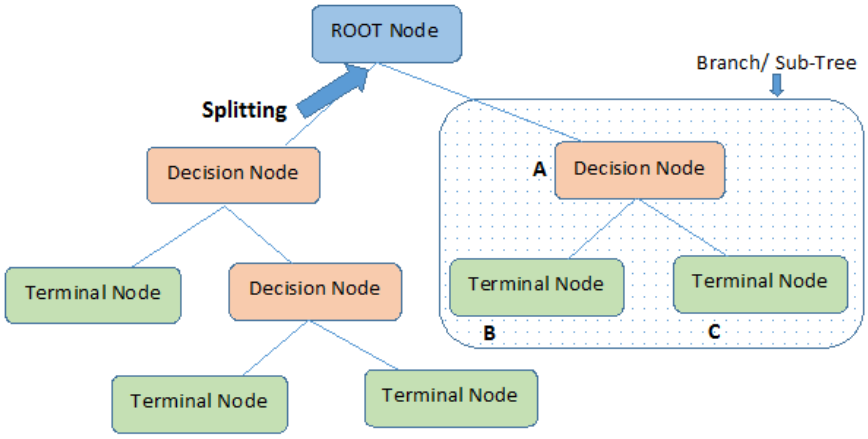


Figure 4.2: Albero di decisione.[2]

Gli alberi di decisione hanno il vantaggio della semplicità. Infatti è possibile seguire il percorso radice-foglia per analizzare come la macchina è giunta a una determinata decisione basandosi sui dati in input. Questo processo rappresenta di fatto la logica utilizzata dalla macchina per interpretare i dati.

Un problema degli alberi di decisione è che tendono ad andare in overfitting.

[2]Fonte: `www.medium.com/datadriveninvestor/decision-tree-algorithm`.

4.2 Naive Bayes

L'algoritmo di classificazione Naive Bayes si basa sull'applicazione del teorema di Bayes:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Contestualizzato per la creazione di un modello per un task di classificazione, si ha la tupla composta dalle features di un particolare documento identificato da:

$$\mathbf{x} = (x_1, ..., x_n)$$

Applicando il teorema di Bayes, rappresentando y come la classe a cui appartiene un individuo, si ottiene:

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x} \mid y)P(y)}{P(\mathbf{x})}$$

Si noti che la parte destra dell'uguaglianza è la probabilità a priori, ed è stimabile utilizzando il dataset a disposizione. In particolare il numeratore rappresenta la probabilità:

$$P(\mathbf{x} \cap y) = P(x_1, ..., x_n, y)$$

Viene sviluppata la probabilità congiunta utilizzando la chain rule per l'applicazione ripetuta della porbabilità condizionata:

$$\begin{aligned} P(x_1, ..., x_n, y) &= P(x_1 \,|\, x_2, ..., x_n, y)P(x_2, ..., x_n, y) \\ &= P(x_1 \,|\, x_2, ..., x_n, y)P(x_2 \,|\, x_3, ..., x_n, y)P(x_3, ..., x_n, y) \\ &= ... \\ &= P(x_1 \,|\, x_2, ..., x_n, y)P(x_2 \,|\, x_3, ..., x_n, y)...P(x_{n-1} \,|\, x_n, y)P(x_n \mid y)P(y) \end{aligned} \tag{4.1}$$

Il classificatore preso in questione è considerato "naive", in quanto si assume che le features siano indipendenti tra loro, pertanto:

$$P(x_i \,|\, x_{i+1}, ..., x_n, y) = P(x_i \mid y)$$

Si procede quindi sviluppando l'equazione per il calcolo della probabilità a posteriori:

$$P(y \mid x_1, x_2, ..., x_n) = \frac{P(x_1 \mid y)...P(x_n \mid y)P(y)}{P(x_1)...P(x_n)}$$

A questo punto, date le features di un individuo, è possibile calcolare la probabilità che questo appartenga ad una particolare classe y. Inoltre, fissate le fetures, il denominatore è costante. Si considera quindi:

$$P(y \mid x_1, x_2, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

Un individuo viene classificato come quella classe y che massimizza la probabilità a posteriori:

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

4.3 Multinomial Naive Bayes

Nell'algoritmo Naive Bayes, avendo dati n-dimensionali e k classi, la probabilità a posteriori si ottiene conoscendo $P(x_i \mid y_j)$ per ogni $1 \leq n,\ 1 \leq j \leq k$

Queste probabilità vanno stimate dal dataset, e nella variante Multinomial Naive Bayes si considera una distribuzione binomiale in più variabili (da qui multinomiale). Si hanno quindi i parametri $\theta_y = (\theta_{y1}, \theta_{y2}, ...\theta_{yn})$, dove $\theta_{yi} = P(x_i \mid y)$ vengono stimati in questo modo:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

Dove, indicando con T il training set:

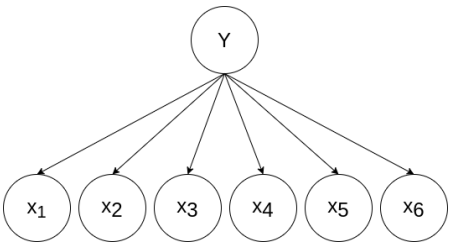
$$N_{yi} = \sum_{x \in T} x_i \qquad N_y = \sum_{i=1}^{n} N_{yi}$$

Il parametro $\alpha : 0 < \alpha \leq 1$ è chiamato di smoothing, e ha lo scopo di gestire quei casi in cui una features non è presente nel training set, così da non ottenere probabilità uguali a 0.

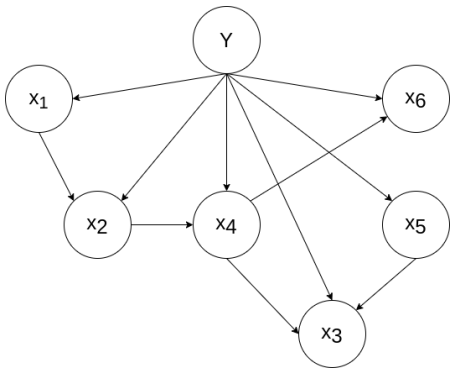
4.4 Bayesian Network

I classificatori naive Bayes si basano sull'assunzione di indipendenza delle caratteristiche. Ciò permette di semplificare le cose quando si considera l'equazione (4.1) come risultato dell'applicazione della chain rule. Tuttavia questa è un'assunzione piuttosto forte.

A questo scopo i classificatori Bayesian Network introducono una struttura dati per far fronte a questo problema. Viene pertanto considerato un DAG (Directed acyclic graph) in cui le n features compongono i nodi del grafo, e gli archi rappresentano la dipendenza tra le variabili.



(a) Naive Bayes network



(b) Bayes network per problema specifico

Figure 4.3: Dipendenza tra variabili.

Considerando le n features indipendenti tra loro si ha:

$$P(X_1, ..., X_n) = \prod_{i=1}^{n} P(X_i)$$

Senza questa assunzione si applica la chain rule per la probabilità congiunta, la quale è computazionalmente onerosa:

$$P\left(\bigcap_{i=1}^{n} X_i\right) = \prod_{i=1}^{n} P\left(X_i \middle| \bigcap_{j=1}^{i-1} X_j\right)$$

Tuttavia si sfrutta il grafo diretto aciclico G così definito:

- $G = (V, E)$
- $V = \{X_1, ..., X_n\}$
- $E = \{(u, v) \mid (u, v) \in V^2 \wedge P(u, v) \neq P(u)P(v)\}$

Il quale rappresenta le dipendenze tra variabili. Quindi nello sviluppo della probabilità congiunta, si considerano solo quei fattori con un arco presente nel grafo. Pertanto, indicando con $\pi(u) = \{v \mid (v, u) \in E\}, v, u \in V$ si ha:

$$P(X_1, ..., X_n) = \prod_{i=1}^{n} P(X_i \mid X_1, ..., X_{i-1}) = \prod_{i=1}^{n} P(X_i \mid \pi(X_i))$$

Così facendo viene semplificata la probabilità congiunta, pur non assumendo la mutua indipendenza tra le variabili.

4.5 Support Vector Machine

Support vector machine è un'altro algoritmo di machine learning supervisionato, in cui gli n individui sono rappresentati nella forma:

$$(\vec{x}_1, y_1), ..., (\vec{x}_n, y_n)$$

Dove $y_i \in \{1, -1\}$ rappresenta la classe di appartenenza di un individuo. Ogni $\vec{x}_i$ è un vettore reale k-dimensionale, con tante dimensioni quante sono le features.

4.5.1 Iperpiano

L'obbiettivo dell'algoritmo è trovare l'iperpiano ottimo che che meglio separa le due classi. Nel caso 2-dimensionale, l'iperpiano è individuato da una retta, tuttavia è facilmente generalizzare il concetto al caso 3-dimensionale e così via:

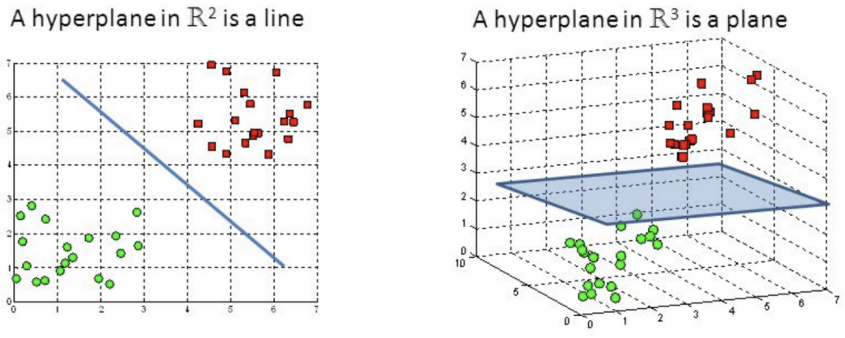


Figure 4.4: Iperpiano ottimo.[3]

4.5.2 Iperpiano ottimo

Per iperpriano ottimo si intende quello che massimizza la distanza tra l'iperpiano e il più vicino punto $\vec{x}_i$ di ciascuna classe. Indicando con $\vec{w}$ il vettore normale all'iperpiano, ogni iperpiano può essere scritto come l'insieme dei punti $\vec{x}$ che soddisfano l'equazione:

$$\vec{w} \cdot \vec{x} - b = 0$$

Vengono considerati i due iperpiani paralleli che separano le due classi, la quale distanza tra i due è la più grande possibile. La regione compresa tra questi è detta margine, e l'iperpiano ottimo si trova nel mezzo tra i due. I due iperpiani possono essere scritti come:

$$\vec{w} \cdot \vec{x} - b = 1$$

$$\vec{w} \cdot \vec{x} - b = -1$$

Essendo $\vec{w}$ il vettore normale all'iperpiano, la distanza tra i due è $\frac{2}{||\vec{w}||}$, pertanto per massimizzare la distanza si vuole minimizzare $||\vec{w}||$.

Si considerano inoltre i vincoli per esprimere che ogni punto deve cadere nel lato corretto rispetto al margine:

$$\vec{w} \cdot \vec{x} - b \geq 1, \ se \ y_i = 1$$

$$\vec{w} \cdot \vec{x} - b \leq -1, \ se \ y_i = -1$$

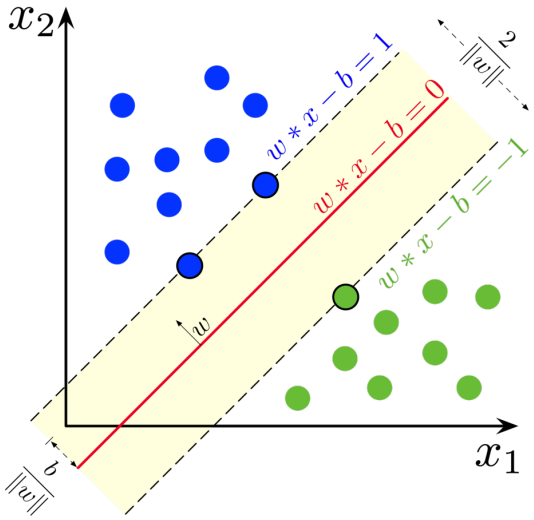


Figure 4.5: Margine SVM.[4]

4.5.3 Vettori di supporto

I due iperpiani che individuano il margine, sono trovati a partire dai vettori che appartengono al dataset. Quei punti che permettono di individuare le frontiere del margine, ovvero i due iperpiani

[3]Fonte: www.towardsdatascience.com/a-friendly-introduction-to-support-vector-machines-svm-925b68c5a079.

[4]Fonte: www.wikipedia.org/wiki/File:SVM_margin.png.

paralleli, sono i vettori di supporto. Questi punti del dataset sono fondamentali, in quanto per costruzione cadono sul bordo del margine, pertanto lo spostamento o la rimozione di uno di essi, avrebbe come risultato due iperpiani diversi che definiscono il margine, quindi un differente iperpiano ottimo.

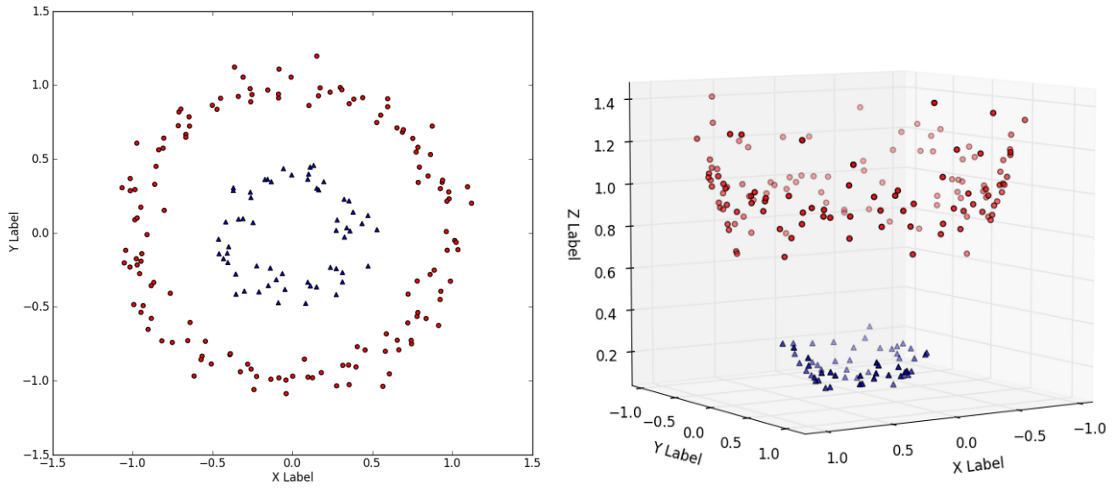
4.5.4 SVM non lineare

Fino ad ora sono stati presi in considerazione solo dataset linearmente separabili. Tuttavia questo può non essere sempre possibile. L'idea è quindi di mappare i dati in uno spazio con una dimensione in più, con lo scopo di ottenere la separazione lineare utilizzando un iperpiano nello spazio con una dimensione aggiuntiva.

Sotto viene preso come esempio un dataset in R^2 non separabile utilizzando un iperpiano. Si considerano quindi la trasformazione:

$$\phi\colon R^2 \to R^3$$
$$\phi(\boldsymbol{X}) = [x_1, x_2, x_1^2 + x_2^2]^T$$

Utilizzando tale trasformazione, il dataset di partenza viene mappato in uno spazio con una dimensione in più:



(a) Dataset in R^2

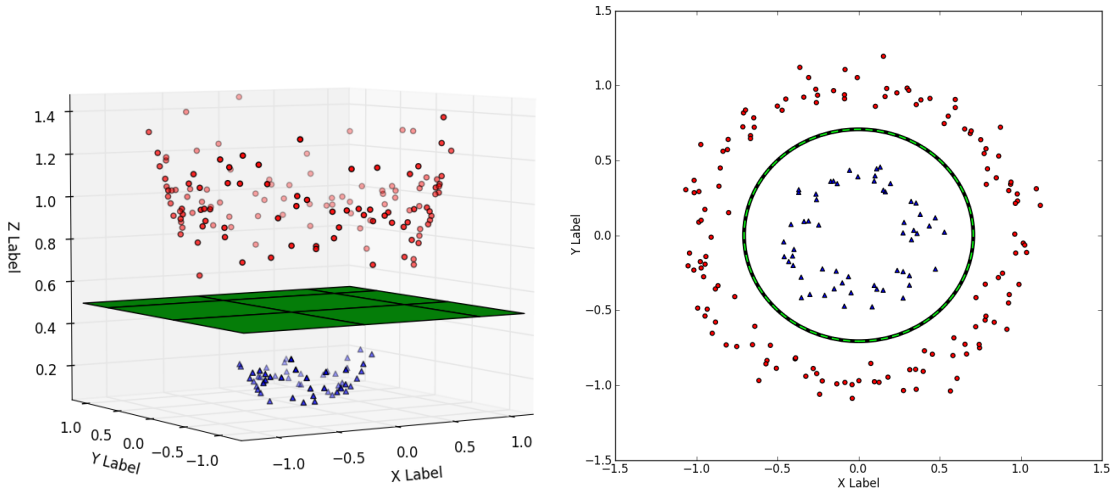
(b) Dataset trasformato in R^3

Figure 4.6: Aggiunta di una nuova dimensione.[5]

Dopo la trasformazione, come si può notare dalla figura 4.6b, è possibile trovare un iperpiano in R^3 che ben separa le due classi:
Nella figura 4.7b, è riportata la linea di dicisione per le due classi, ovvero l'iperpiano trovato in R^3 e poi proiettato in R^2.

[5]Fonte: www.medium.com/@vivek.yadav/why-is-gradient-descent-robust-to-non-linearly-separable-data-a50c543e8f4a.

[6]Fonte: https://www.eric-kim.net/eric-kim-net/posts/1/imgs/data_2d_to_3d_hyperplane.png.



(a) Iperpiano in R^3

(b) Iperpiano proiettato in R^2

Figure 4.7: Aggiunta di una nuova dimensione.[6]

4.6 Strumenti utilizzati

Per l'implementazione degli algoritmi, sono stati utilizzati degli strumenti che mettono a disposizione la possibilità di effettuare il training dei modelli.

4.6.1 Scikit-learn

Scikit-learn[7] è un modulo di python open soruce per il machine learning. Utilizzando questo modulo, è stato scritto il codice python per creare i modelli con gli algoritmi:

- Decision Tree
- Multinomial Naive Bayes
- Sipport Vector Machine - Kernel lineare
- Sipport Vector Machine - Kernel RBF

Inoltre per ogni algoritmo sono stati creati più modelli considerando diverse l'insieme potenza delle features. Quindi sono stati eseguiti i seguenti macrostep:

1. Lettura del dataset
2. Shuffle del dataset con lo stesso seed per ogni modello
3. Standardizzare del dataset
4. Creazione di 10 folds in cui viene effettuato lo split del dataset tra training-set e test-set
5. Training dei modelli per ogni folds

[7]Scikit-learn: www.scikit-learn.org/stable/

6. Valutazione dei classificatori per ogni folds
7. Media delle misure di performance rispetto ogni folds
8. Dump del report così da poter essere sucessivamente messo a confronto con altri modelli

Essendo scikit-learn un modulo per python, che mette a disposizione delle API, è stato possibile gestire manualmente la politica per il training.

Infatti per questioni di ottimizzazione, fissato un algoritmo e le features per un determinato modello, è stato scelto di creare k processi eseguiti in parallelo, ognuno dei quali si prende in carico il training del modello su un particolare fold dei k considerati. Avendo scelto per gli esperimenti 10 folds, ed utilizzato una macchina con 8 cores, ciò si presta bene sfruttare ognuno degli 8 cores, con il risultato di ridurre il tempo totale impiegato per il training dei diversi modelli, dovendo considerare diverse features per ogni modello e i molteplici algoritmi di classificazione.

4.6.2 Weka

Weka[8] è un software open source per il machine learning. Questo applicativo può essere utilizzato attraverso interfaccia grafica, ed è in generale più immediato rispetto a scrivere del codice in python che utilizza le API fornite dal modulo scikit-learn. Con questo software sono stati creati i modelli utilizzando l'algoritmo:

- Bayesian network

Non è stato utilizzato scikit-learn in quanto questo algoritmo non è implementato nel modulo.

Essendo weka un software che fornisce un'interfaccia grafica, ha il vantaggio di essere più immediato rispetto a scikit-learn, tuttavia non essendo possibile implementare manualmente lo script per il training, in questo caso non vengono sfruttati tutti i cores per il training dei modelli.

[8] Weka: www.cs.waikato.ac.nz/ml/weka/

Chapter 5

Campagna sperimentale

5.1 Dataset

Il dataset a disposizione è il corpora *TwReyes2013*, composto da 40,000 tweets in lingua inglese. Trattandosi di un approccio supervisionato, è richiesto che il dataset sia etichettato. Per associare una specifica label ai messaggi generati dagli utenti si possono seguire due strade:

- **Self-Tagging**: Twitter mette a disposizione l'utilizzo degli hashtag nei messaggi. Assumendo che un utente utilizzi l'hashtag *#irony* con la volontà di esprimere ironia, è facile collezionare una serie di tweet etichettati come ironici.
- **Crowdsourcing**: I vari tweet vengono etichettati manualmente da alcune persone

Nel caso specifico, viene utilizzato il dataset *TwReyes2013*, composto da 40,000 tweet in lingua inglese, accumulati usando la tecnica self-tagging. Vengono quindi considerati 4 hashtag diversi:

Numero	**Hashtag**	**Label assocaita**
10,000	*#irony*	ironico
10,000	*#education*	non ironico
10,000	*#humor*	non ironico
10,000	*#politics*	non ironico

Table 5.1: Hashtag e label associate

5.2 10-Folds Cross-Validation

Per la fase di training dei modelli è necessario suddividere il dataset a disposizione in due diversi insiemi: training-set e test-set. Questo è un passaggio fondamentale in quanto non si otterrebbero delle misure di performance valide testando i modelli su dati che sono stati utilizzati per la fase di trainig. In pratica si vuole testare il modello su dati che non sono mai stati visti in precedenza.

Prima di iniziare la fase ti training, i dati in input, che sono composti dalla matrice contenente le features estratte, vengono opportunamente mescolati. Con questo si intende che le righe della matrice vengono casualmente cambiate di posto, chiaramente mantenendo coerenza con le labels

associate. Un aspetto importante è che lo shuffle della matrice è eseguito con lo stesso seed per tutti i modelli, così da ottenere delle misure di performance statisticamente valide.

Le performance ottenute, dipendono anche da come è stato diviso il dataset tra training-set e test-set. Per questo motivo viene utilizzata la tecnica k-Folds Cross-Validitation, che permette di sfruttare diverse porzioni di dati per i due insiemi. Il vantaggio di questa tecnica è quello di ottenere valutazioni più affidabile mediando i risultati di ognuno dei k modelli di cui è stato fatto il training, a discapito del tempo necessario, in quanto si effettuerà k volte il training di un modello.

Per gli esperimenti effettuati, è stato scelto $k = 10$, ovvero il dataset è stato diviso 10 volte, utilizzando ad ogni iterazioni una porzione di dati diversa per il trainig-set e test-set. Avendo scelto questo valore di k, la proporzione tra i due insiemi sarà sempre 90% per il training-set e 10% per il test-set.

Nel seguente diagramma si possono considerare i pallini verdi e rossi come i tweets ironici e non ironici. Quindi, fissata un'istanza del dataset mescolato, si considera una sliding window di dimensione $\#samples \cdot \frac{1}{k}$, dove per ogni k iterazione viene effettuato il training di un nuovo modello.

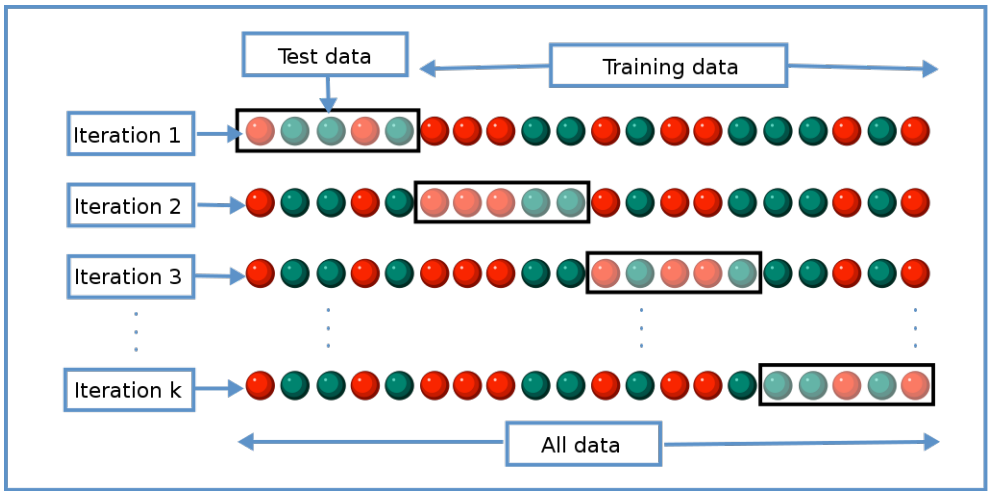


Figure 5.1: k-Folds Cross-Validation.
`www.wikipedia.org/wiki/Cross-validation_(statistics).`

5.3 Misure di performance

Essendo stati creati diversi modelli, è necessario valutarli. Avendo diviso il dataset a disposizione tra test-set e training-set, una volta completata la fase di training è possibile utilizzare il test-set per la valutazione. Si procede prendendo un tweet e facendolo classificare al modello, il quale verrà quindi identificato come ironico o non ironico. Trattandosi di modelli supervisionati, oltre ai dati si hanno a disposizione le labels associate, è quindi possibile confrontare l'output fornito dal modello con la reale classe di un tweet. Da qui si può capire se per quel particolare individuo, la classificazione è corretta o meno.

Per la valutazione si utilizza tutto il test-set a disposizione e viene contano il numero di:

- **TP** (True positive): Un tweet ironico è classificato come ironico - *corretto*

- **TN** (True negative): Un tweet non ironico è classificato come non ironico - *corretto*
- **FP** (False positive): Un tweet non ironico è classificato come ironico - *errato*
- **FN** (False negative): Un tweet ironico è classificato come non ironico - *errato*

Si noti che un modello può sbagliare la classificazione in due modi diversi, e c'è differenza tra classificare un tweet ironico come non ironico piuttosto che uno non ironico come ironico. Questi valori verranno poi analizzati sfruttando una matrice di confusione:

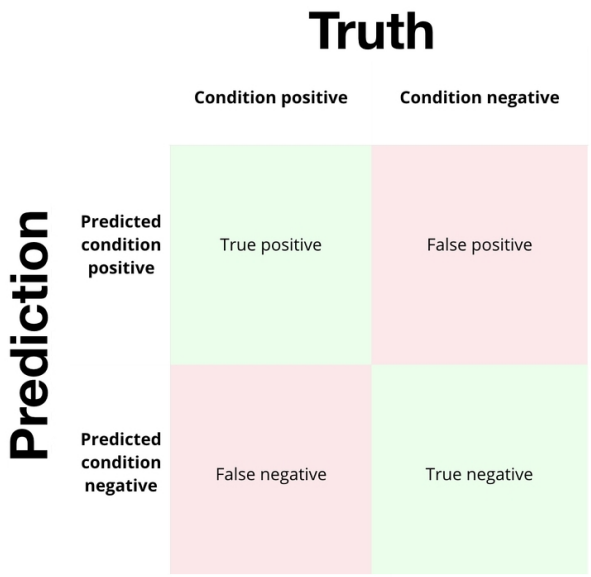


Figure 5.2: Confusion matrix. `www.radiopaedia.org/cases/confusion-matrix-3`.

Basandosi su questi dati vengono definite delle misure di performance per la valutazione:

- **Accuracy**: Il rapporto tra il numero di classificazioni corrette e le classificazioni totali.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

- **Precision**: La porizione di classificazioni positive corrette.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall**: La porzione di sample positivi identificati.

$$Recall = \frac{TP}{TP + FN}$$

- **F-measure**: Metrica per calcolare la valutazione complessiva. Formalmente è definita come la media armonica tra recall e precision.

$$F\text{-}measure = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

5.4 Esperimenti

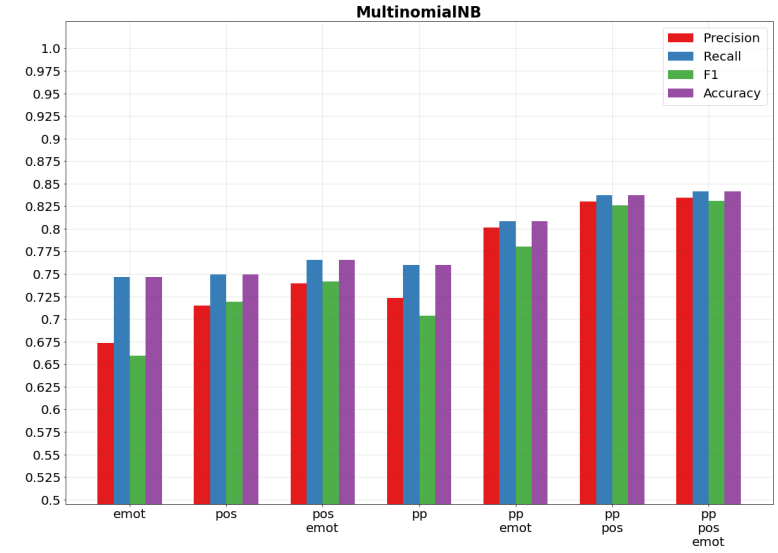
Avendo creato diversi modelli, questi sono stati confrontati per valutare il più performante. Nello specifico sono stati creati i modelli con gli algoritmi di classificazione:

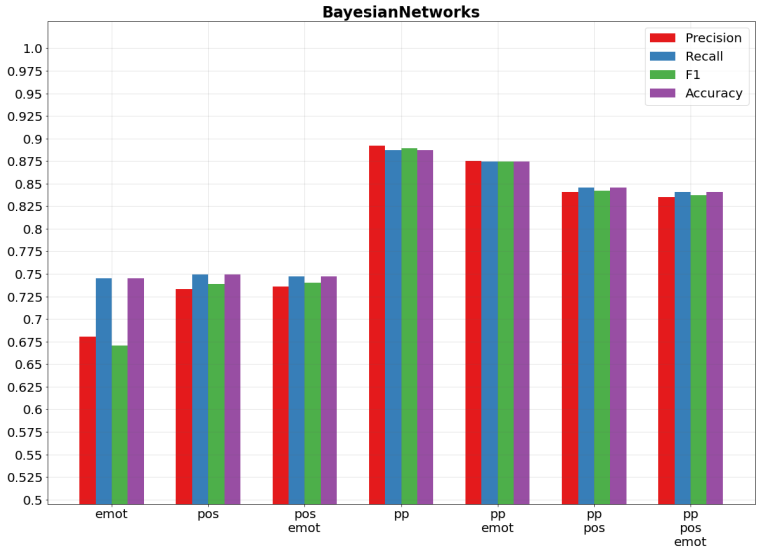
- Decision Tree
- Multinomial Naive Bayes
- Bayesian Network
- Support Vector Machine - Lineare

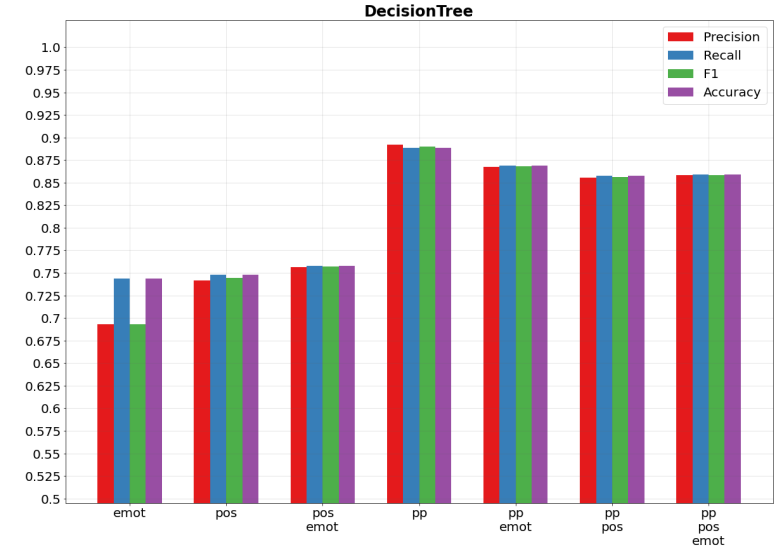
Inoltre sia $F = \{text,\ pp,\ pos,\ emot\}$ l'insieme delle features considerate. Per ogni algoritmo di classificazione, sono stati creati i modelli utilizzando come features $\mathscr{P}(F)$.

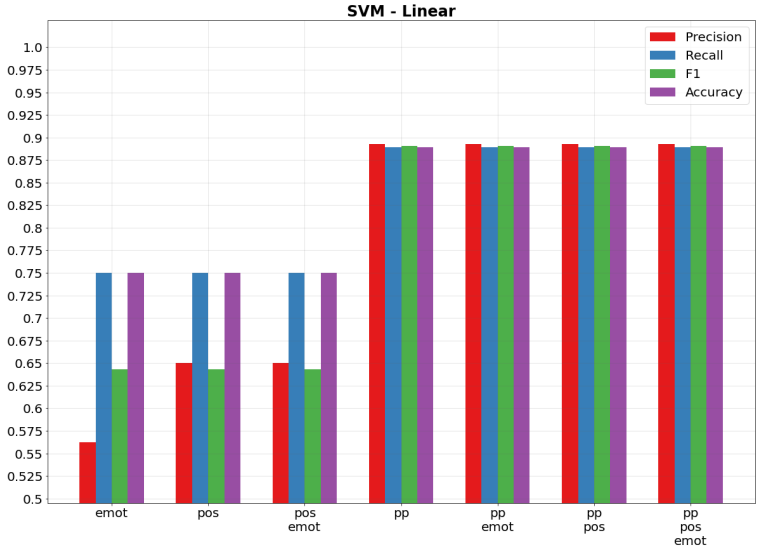
5.4.1 Caratteristiche linguistiche

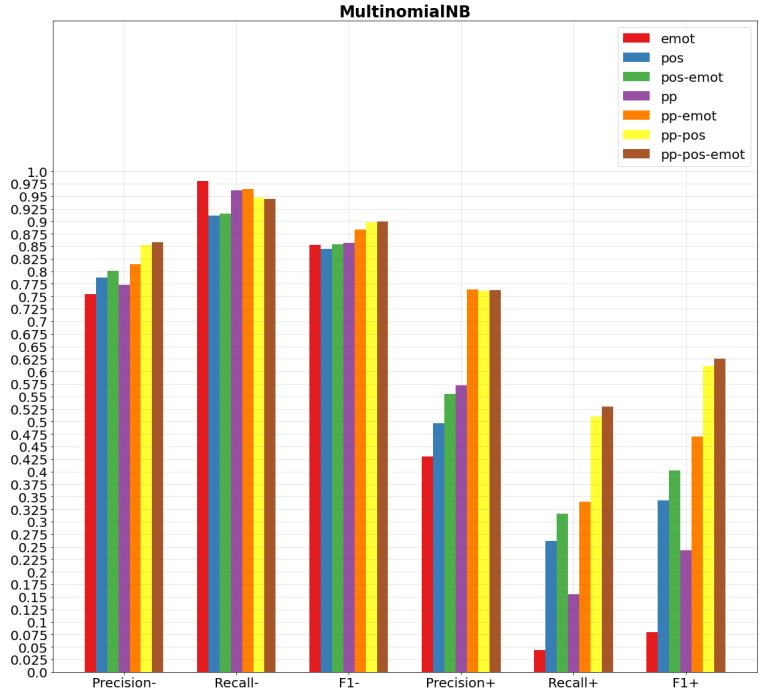
5.4.1.1 Weighted average

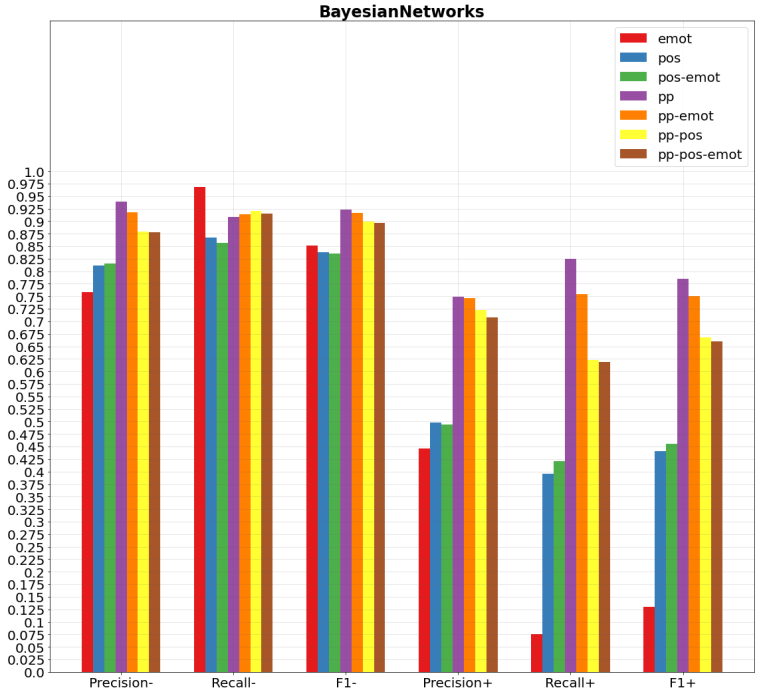


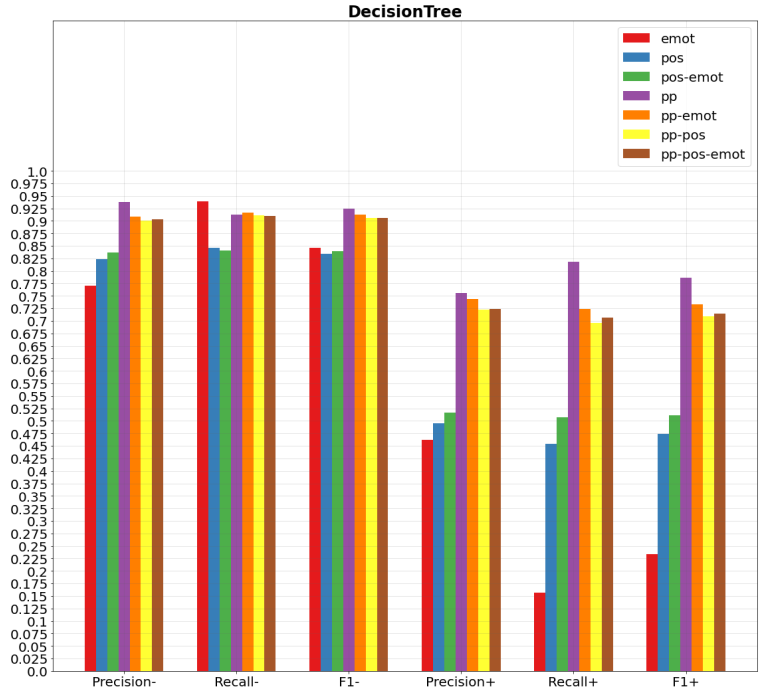


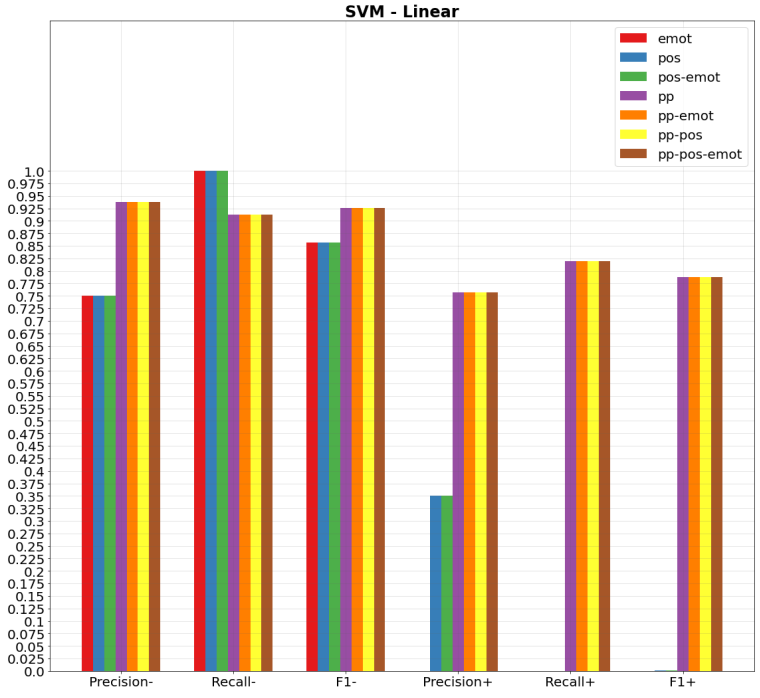




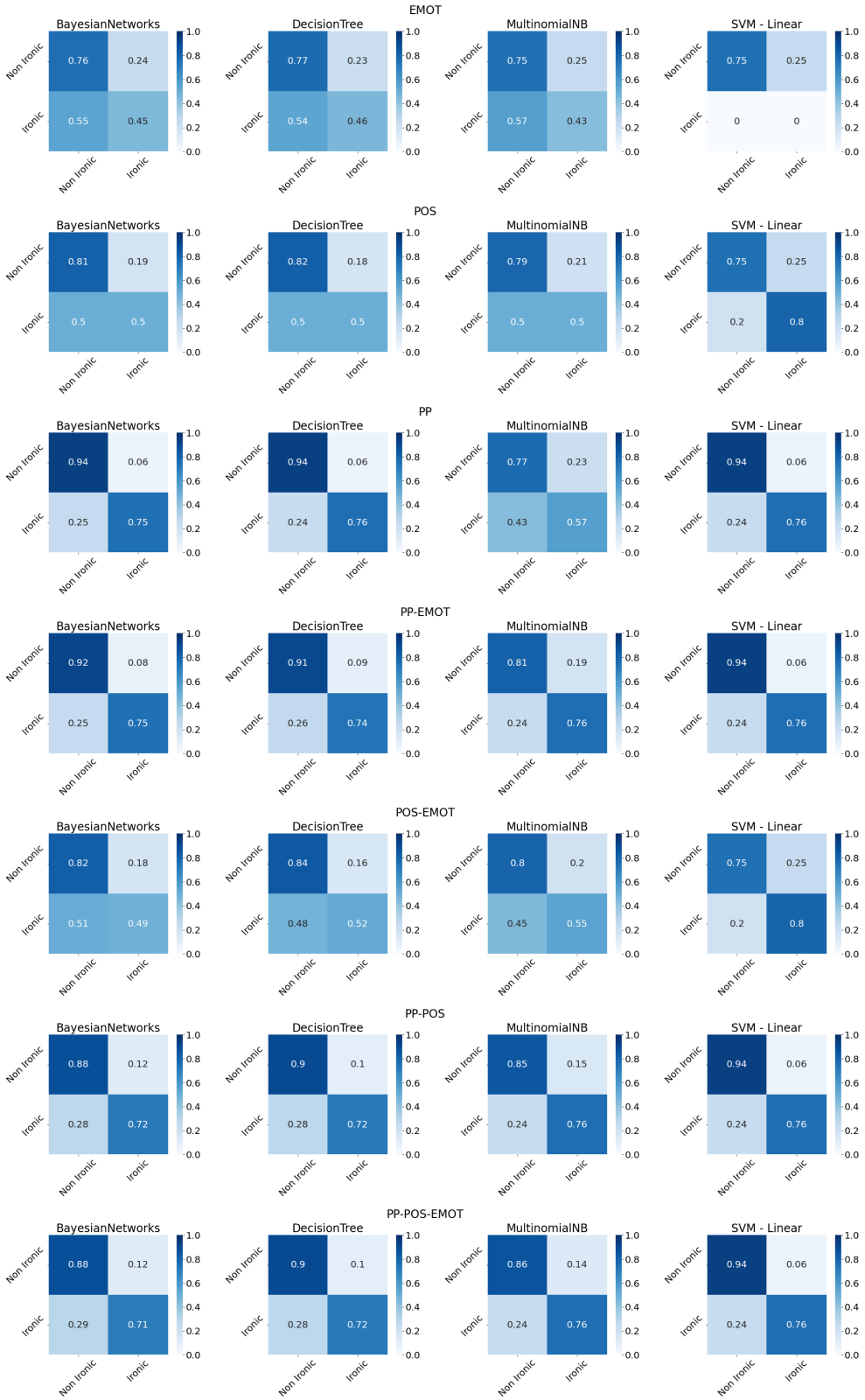

MultinomialNB
emot
pos
pos-emot
pp
pp-emot
pp-pos
pp-pos-emot
Precision-
Recall-
F1-
Precision+
Recall+
F1+


BayesianNetworks
emot
pos
pos-emot
pp
pp-emot
pp-pos
pp-pos-emot
Precision-
Recall-
F1-
Precision+
Recall+
F1+


DecisionTree
emot
pos
pos-emot
pp
pp-emot
pp-pos
pp-pos-emot
Precision-
Recall-
F1-
Precision+
Recall+
F1+

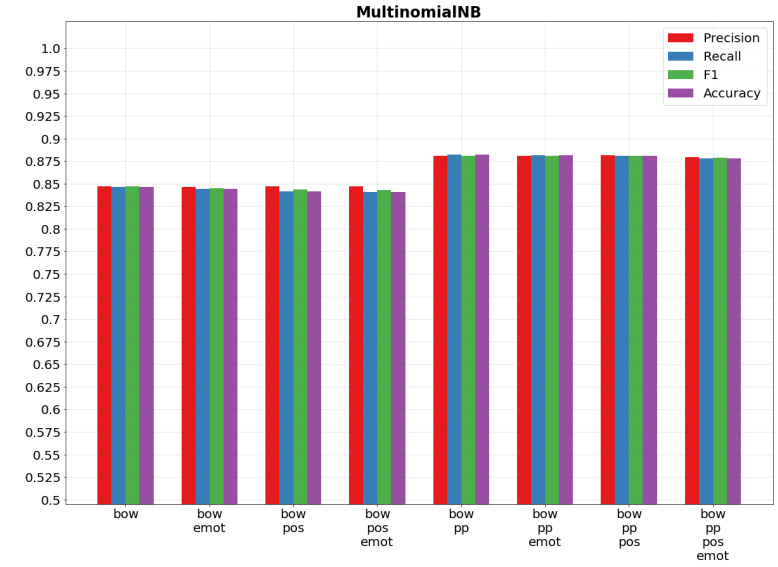

SVM - Linear
emot
pos
pos-emot
pp
pp-emot
pp-pos
pp-pos-emot
Precision-
Recall-
F1-
Precision+
Recall+
F1+

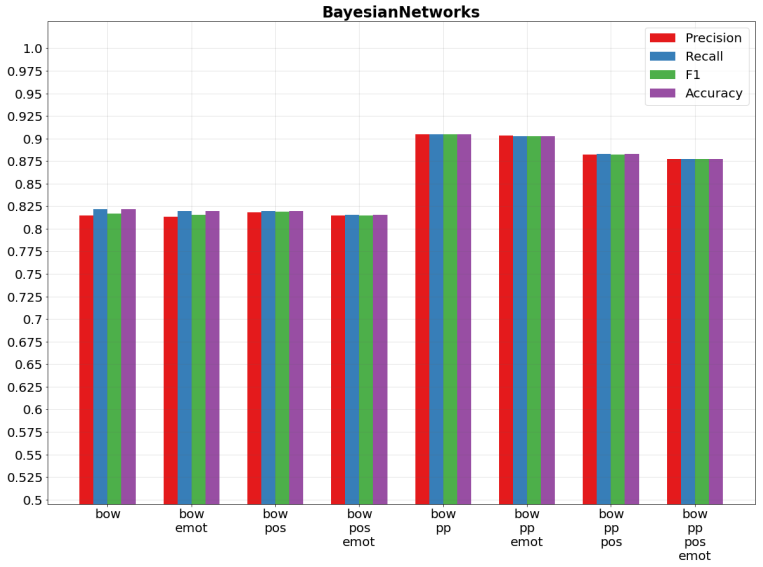
5.4.1.2 Confusion matrix

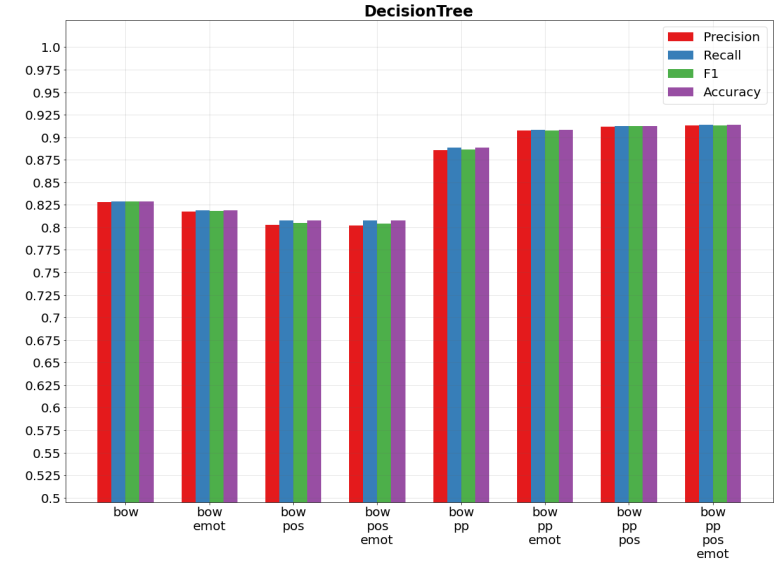


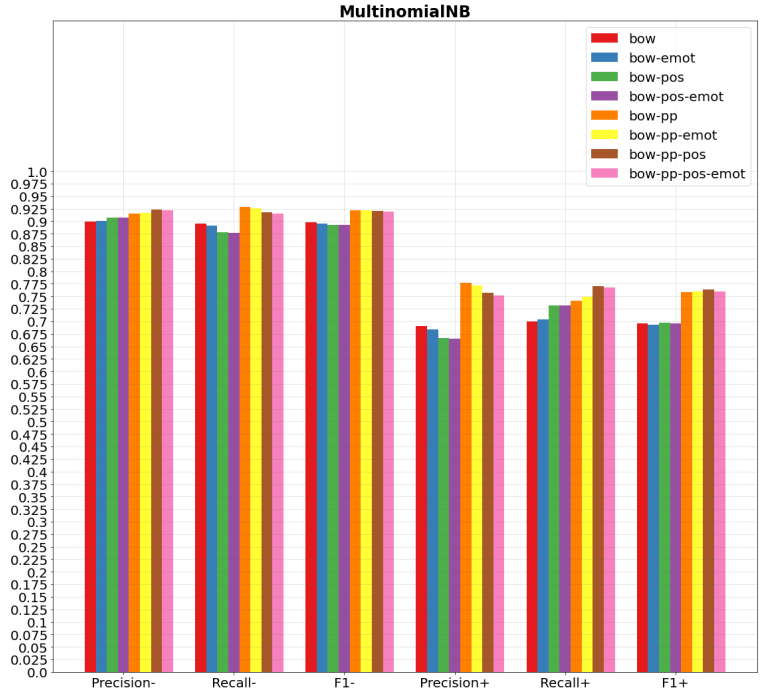
5.4.2 BOW + Caratteristiche linguistiche

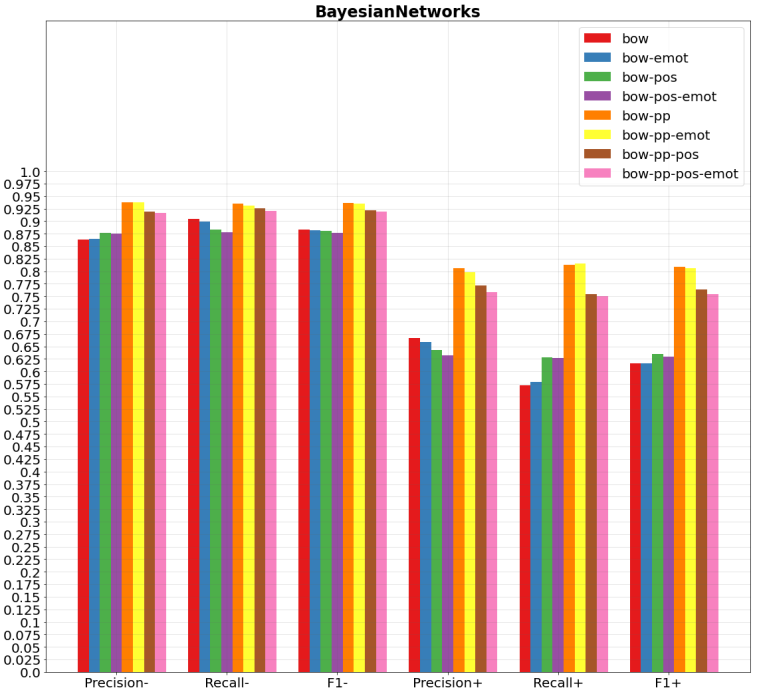
5.4.2.1 Weighted average

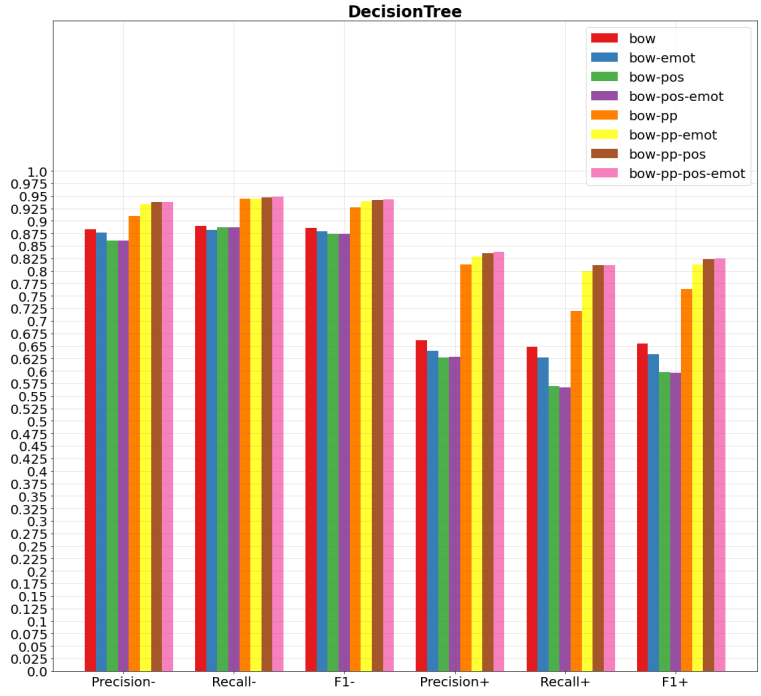


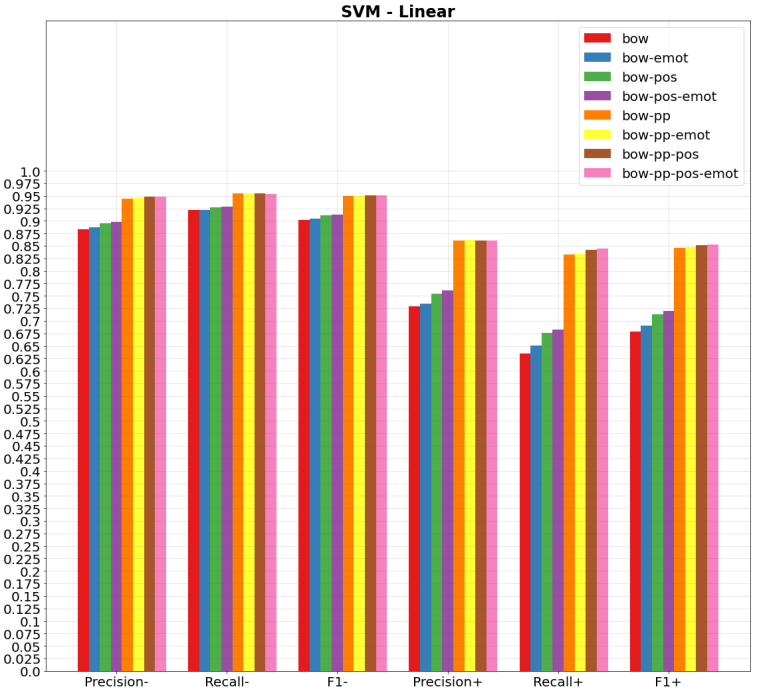




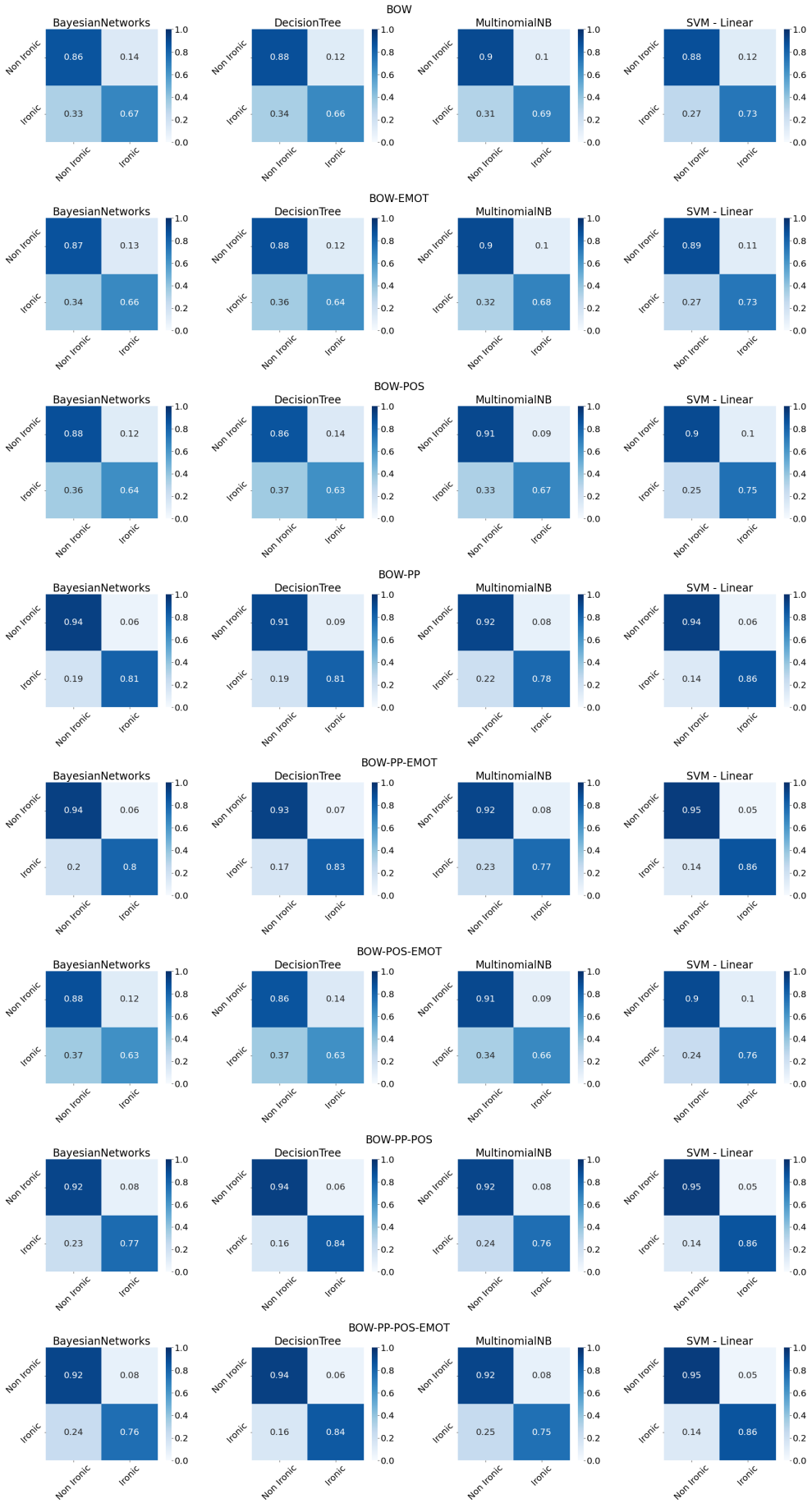

MultinomialNB
bow
bow-emot
bow-pos
bow-pos-emot
bow-pp
bow-pp-emot
bow-pp-pos
bow-pp-pos-emot
Precision-
Recall-
F1-
Precision+
Recall+
F1+


BayesianNetworks
bow
bow-emot
bow-pos
bow-pos-emot
bow-pp
bow-pp-emot
bow-pp-pos
bow-pp-pos-emot
Precision-
Recall-
F1-
Precision+
Recall+
F1+


DecisionTree
bow
bow-emot
bow-pos
bow-pos-emot
bow-pp
bow-pp-emot
bow-pp-pos
bow-pp-pos-emot
Precision-
Recall-
F1-
Precision+
Recall+
F1+

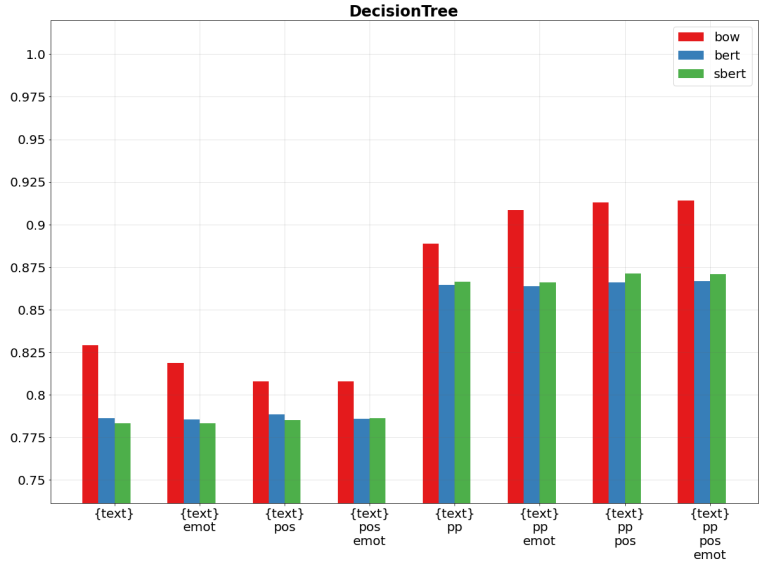

SVM - Linear
bow
bow-emot
bow-pos
bow-pos-emot
bow-pp
bow-pp-emot
bow-pp-pos
bow-pp-pos-emot
Precision-
Recall-
F1-
Precision+
Recall+
F1+

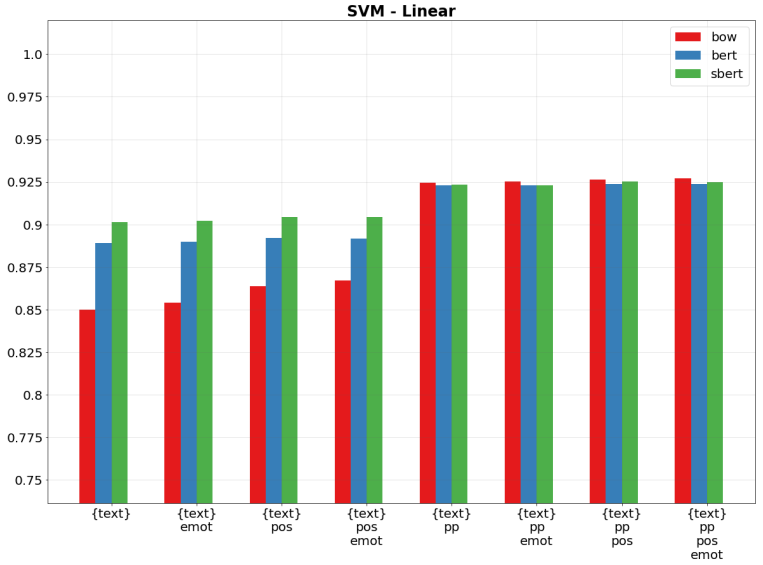
5.4.2.2 Confusion matrix



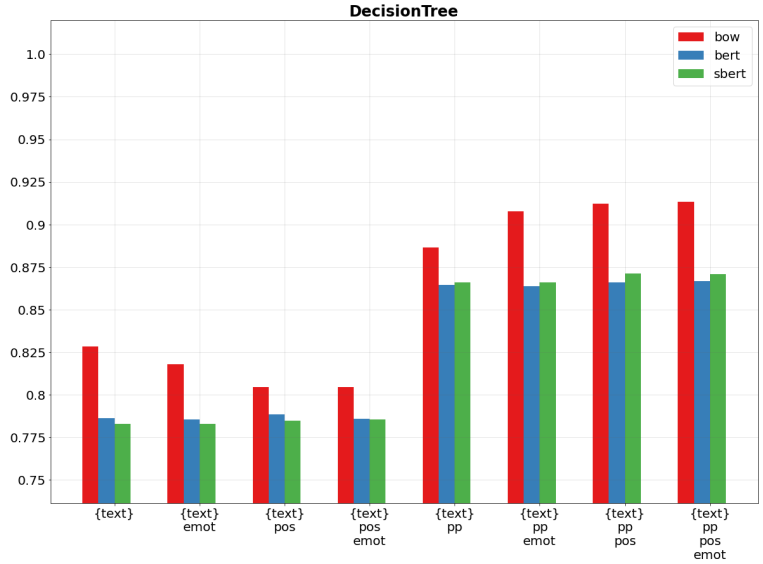
5.4.3 BOW vs BERT vs Sentence-BERT

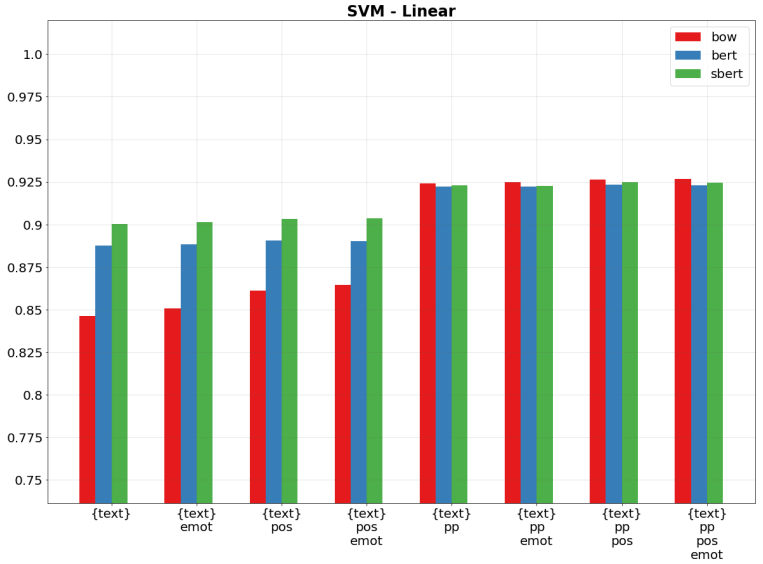
5.4.3.1 Weighted average - Accuracy





5.4.3.2 Weighted average - F measure





5.4.4 Analisi lessico con PCA

Chapter 6

Conclusioni e sviluppi futuri