Sentiment Analysis and Emotion Detection of Italian Tweets on Gender-Based Violence: A Comparative Study of Approaches

Federico Bertoia

15th December 2023

Introduction

Sentiment Analysis and **Emotion Detection** are fields of natural language processing that aim at computationally identifying and categorizing opinions expressed in a piece of text.

Sentiment $\rightarrow$ Negative, Neutral, Positive
Emotion $\rightarrow$ Tristezza, Gioia, Amore, Rabbia, Paura, Sorpresa, Neutra

In this thesis we compared the effectiveness of various approaches:

- **Lexicon-based approach**: VADER
- **Machine Learning approach**: Naïve Bayes Classifier and Support Vector Machines
- **Transformers approach**: BERT

Outline

Outline

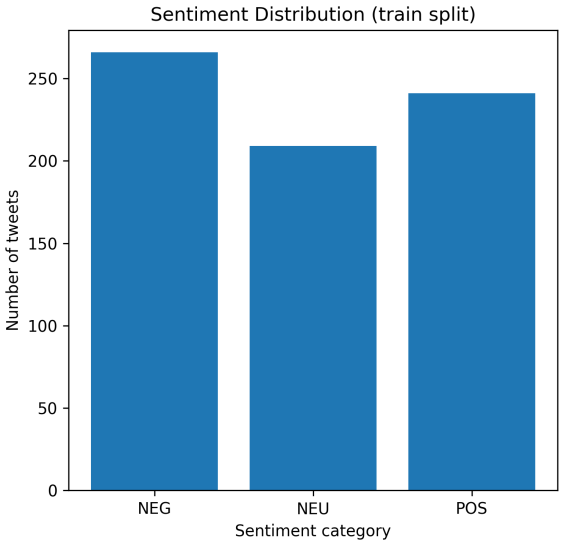
Structure of the Datasets and Visualization

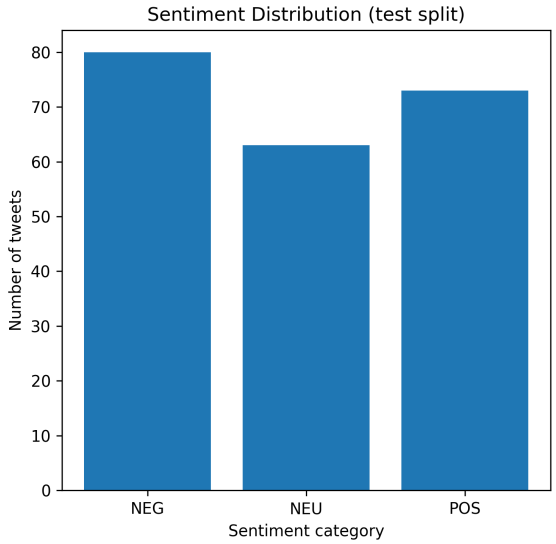
We have a sample of **italian tweets on gender-based violence** provided by ISTAT divided into two datasets.

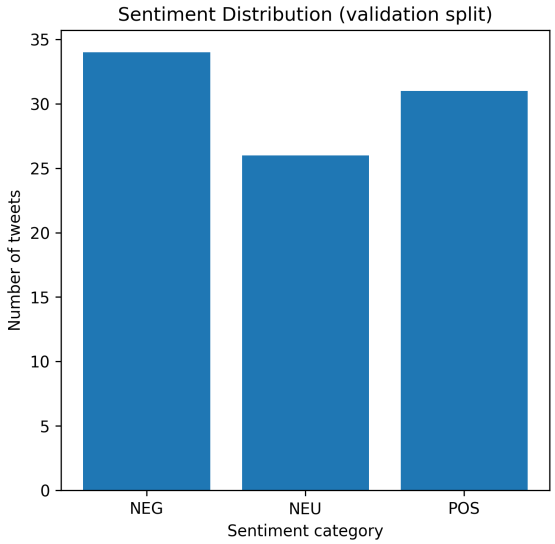
Split	Features	Number of Rows
Train	['sentiment', 'text']	715
Test	['sentiment', 'text']	216
Validation	['sentiment', 'text']	91

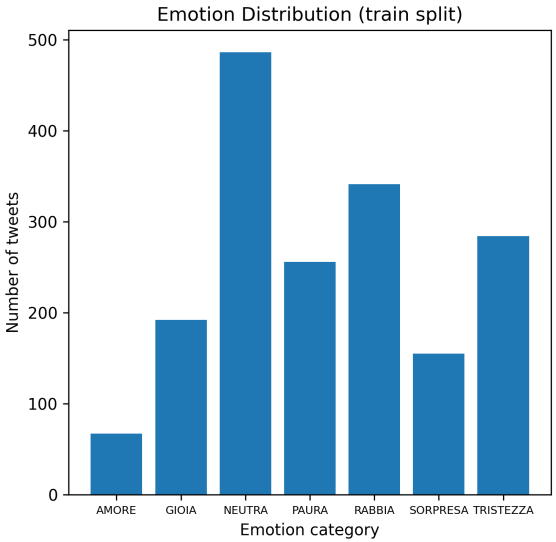
Split	Features	Number of Rows
Train	['emotion', 'text']	1781
Test	['emotion', 'text']	543
Validation	['emotion', 'text']	318

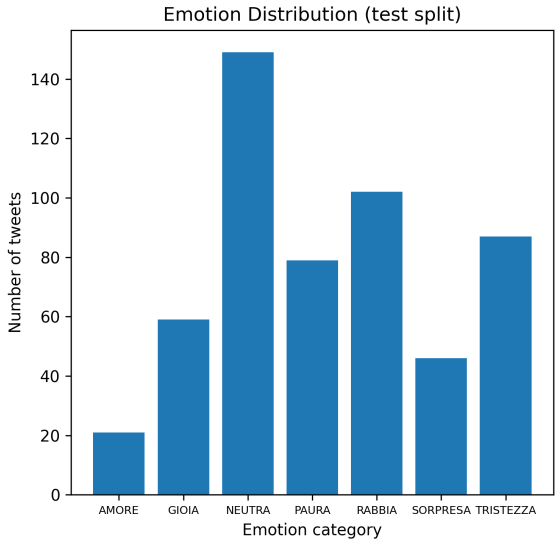
Structure of the Datasets and Visualization (2)

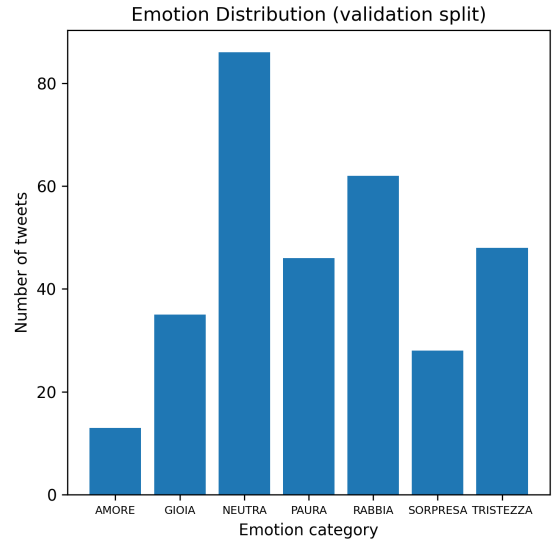












Structure of the Datasets and Visualization (3)

(a) Sentiment Word Cloud

(b) Emotion Word Cloud

Data Preprocessing and Feature Engineering

Data Preprocessing refers to the set of techniques and steps applied to raw text data to transform it into a suitable format for sentiment classification tasks, such as:

- Punctuation removal
- URL removal
- User Mention removal
- Tokenization
- Lemmatization

Feature Engineering involves extracting meaningful information by converting text into numerical representations that can be used as input features for machine learning models. The chosen techniques are:

- Bag of Words ▸ BoW
- TF-IDF ▸ TF-IDF

Evaluation Metrics

Performance indicators are very useful when the aim is to evaluate and compare different classification models. We consider: **accuray**, **precision**, **recall** and **F1-Score**. ▸ Metrics

To get a visual representation we analyze the **confusion matrix**:

		PREDICTED classification				
	Classes	a	b	c	d	Total
ACTUAL classification	a	6	0	1	2	9
	b	3	9	1	1	14
	c	1	0	10	2	13
	d	1	2	1	12	16
	Total	11	11	13	17	**52**

Outline

Lexicon-based Approach: VADER

Lexicons are collections of words where each word is assigned with a predefined score which indicates the **polarity** and **intensity** of the sentiment.

VADER is a lexicon whose scores lie in [-4,4], however it takes into account also the relationship among words by considering punctuation, capitalization, adverbs, conjunctions and negations.

A piece of text is classified based on its **compound score**, which is the normalized score of the sum of valences computed following the lexicon and the heuristics:

- positive sentiment: compound score $\geq$ 0.05
- neutral sentiment: -0.05 $<$ compound score $<$ 0.05
- negative sentiment: compound score $\leq$ -0.05 ▸ VADER

Machine Learning Approach: Naïve Bayes Classifier

Naïve Bayes is a probabilistic classifier thar relies on the **Bayes' Theorem**. Given the class variable c and a document D, Naïve Bayes solves:

$$\hat{c} = \arg\max_{c \in C} P(c \mid D) = \arg\max_{c \in C} \frac{P(D \mid c) \cdot P(c)}{P(D)}$$

Each document can be represented by a feature vector $\mathbf{f} = (f_1, f_2, \cdots, f_p)$, and by assuming conditional independence between every pair of features we get:

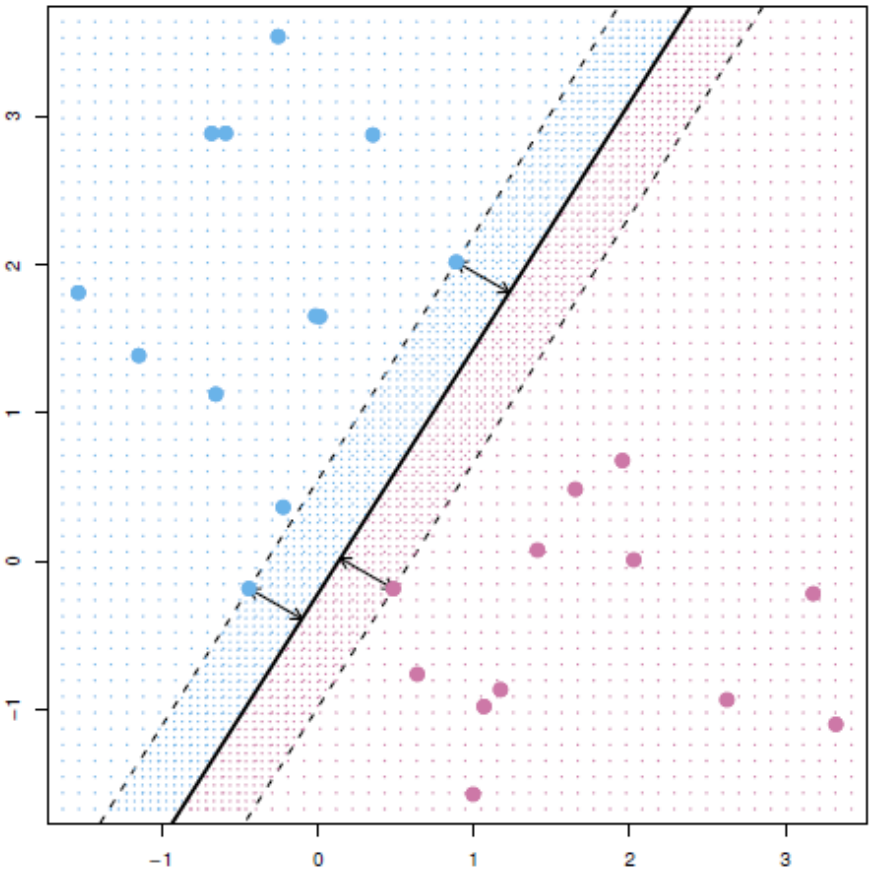
$$c_{\text{NB}} = \arg\max_{c \in C} \left(P(c) \prod_{j=1}^{p} P(f_j \mid c) \right)$$

In the Multinomial Naïve Bayes, we have:

$$P(f_j \mid c) = \hat{\theta}_{cj} = \frac{N_{cj} + \alpha}{N_c + \alpha n}$$

Machine Learning Approach: Support Vector Machines

Support Vector Machines are a class of supervised learning techniques that aims at finding the hyperplane that best separates the data into distinct classes.

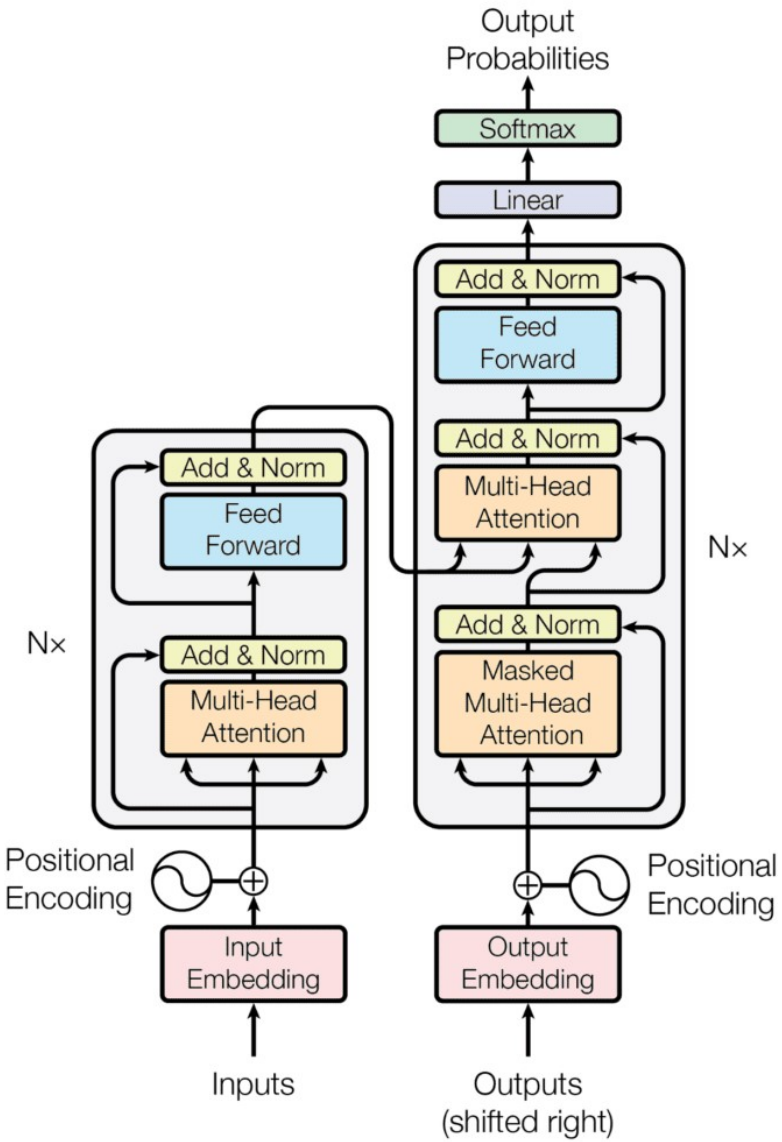


In NLP tasks, we usually have non-linear boundaries; thus, we use kernels to enlarge the feature space. In this new setup, the decision boundary could indeed be linear.

▸ SVM

Outline

Transformers Architecture



The **encoder** is responsible for processing and encoding the input sequence. The **decoder** is responsible for generating the output sequence based on the encoded input.

The Transformer model chosen to perform sentiment analysis and emotion detection is **BERT**, which relies entirely on the encoder part.

▸ BERT

BERT Encoding Pipeline: Tokenization

Tokenization is the process of breaking down a piece of text into smaller units called **tokens**, and assigning a numerical value to each token.

▸ WordPiece

La lotta contro il bodyshaming è una cosa, promuovere l'obesità è un'altra

↓

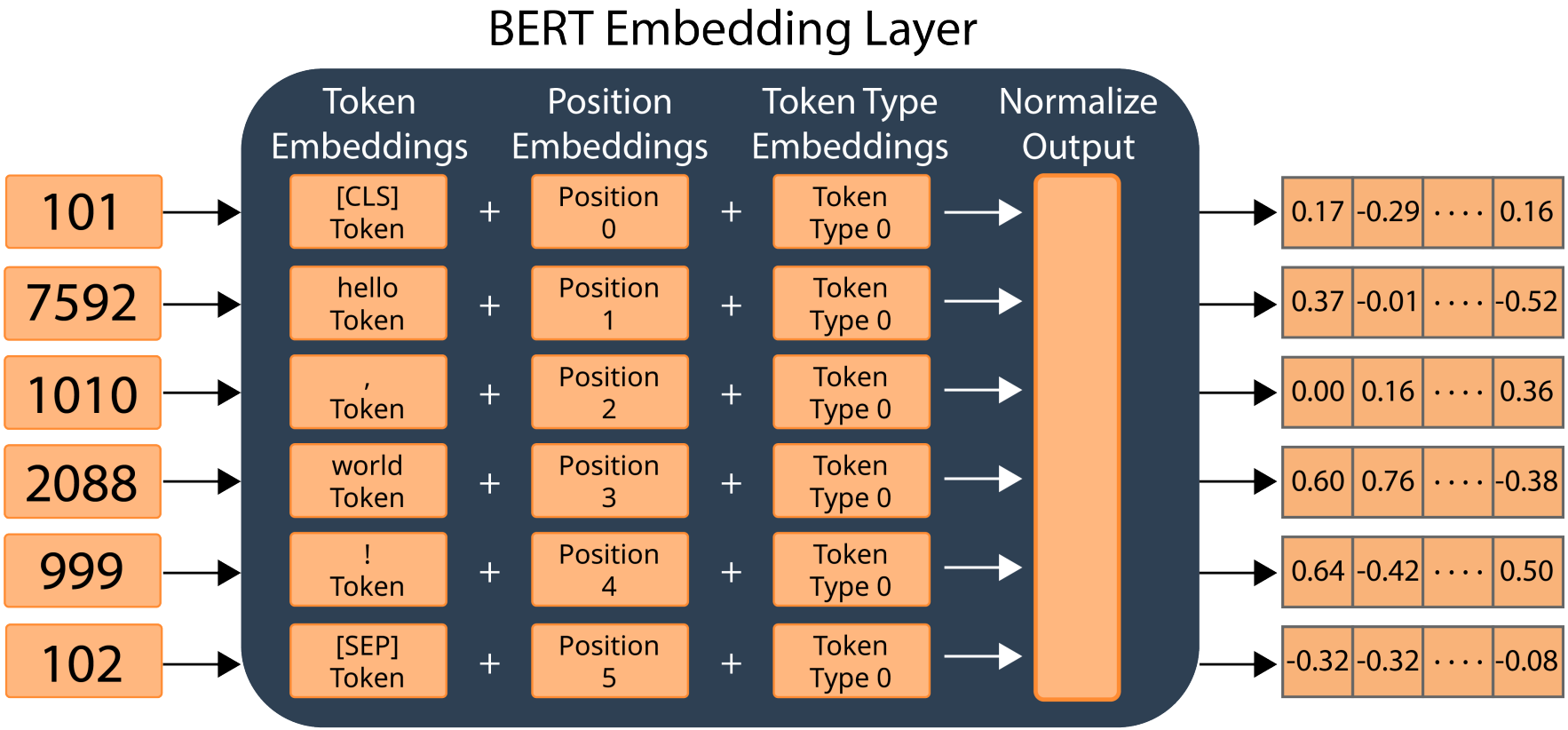
['[CLS]', 'La', 'lotta', 'contro', 'il', 'body', '##sha', '##ming', 'è', 'una', 'cosa', ',', 'promu', '##overe', 'l', '[UNK]', 'ob', '##esi', '##tà', 'è', 'un', '[UNK]', 'altra', '[SEP]']

The tokenizer has two outputs:

- **input_ids**: represent the specific IDs of each token.
- **attention_masks**: indicate which tokens in the input sequence should be attended to and which ones should be ignored during the attention mechanism.

BERT Encoding Pipeline: Embeddings

For each unique Token ID, the BERT model contains an embedding that is trained to represent that specific token.



BERT Encoding Pipeline: Encoder Layer

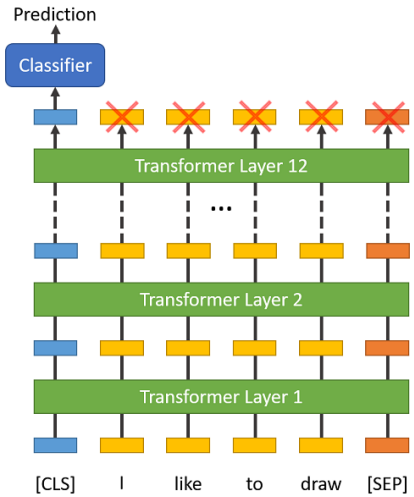
The encoder layers gets as input embeddings, and it returns as output *contextualized* embeddings.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V$$

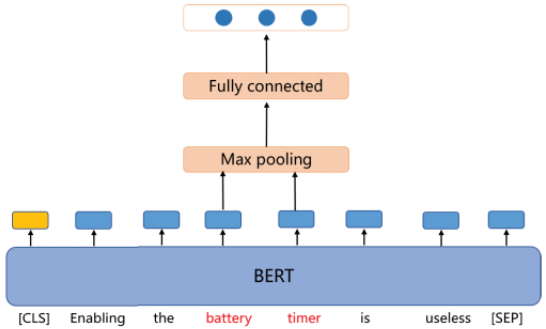
1. **Dot Product of Query and Key**
2. **Softmax of Scaled Scores**
3. **Output and Multi-Head Attention**
4. **Residual Connections, Layer Normalization, and Feed Forward Network**

▸ Attention Mechanism

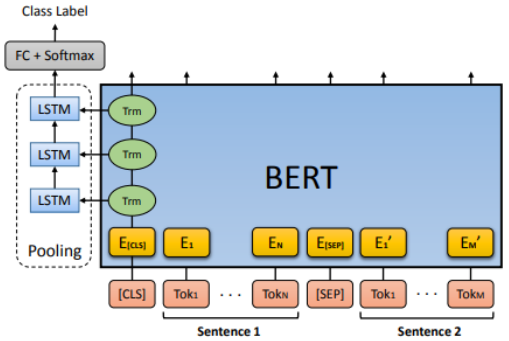
Output Representations



(a) CLS Token



(b) Max Pooling



(c) LSTM Pooling

The **classification head** is built on top of the encoder layer and consists of two key elements: a dense layer and a softmax activation function.

▸ BERT Model Variants

Fine-Tuning

Fine-Tuning is a machine learning technique where a pre-trained model is further trained on a more specific dataset to adapt and optimize its performance for a particular task or problem. ▸ Loss Function and Optimizer

- **Loss Function** → Sparse Categorical Crossentopy
- **Optimizer** → AdamW

Hyperparameter	**Possible Values**
Batch Size	16, 32
Learning Rate (AdamW)	4e-5, 3e-5, 2e-5, 1e-5
Number of Epochs	2, 3, 4

Table: Hyperparameter Values for Grid Search

Outline

Results: Sentiment Analysis

Model	Accuracy	Macro F1	Weighted F1
VADER	0.49	0.44	0.46
Naive Bayes BoW	0.54	0.51	0.52
Naive Bayes TFIDF	0.53	0.47	0.49
SVM BoW	0.55	0.55	0.55
SVM TFIDF	0.55	0.51	0.52
multiBERT-CLS	0.61	0.61	0.62
multiBERT-MAX	0.61	0.60	0.61
multiBERT-LSTM	0.60	0.59	0.59
itaBERT-CLS	0.64	0.64	0.64
itaBERT-MAX	**0.66**	**0.65**	**0.66**
itaBERT-LSTM	0.65	**0.65**	**0.66**

Table: Sentiment Analysis Summary

Results: Emotion Detection

Model	Accuracy	Macro F1	Weighted F1
Naive Bayes BoW	0.71	0.65	0.70
Naive Bayes TFIDF	0.66	0.51	0.62
SVM BoW	0.65	0.62	0.65
SVM TFIDF	0.70	0.66	0.69
multiBERT-CLS	0.79	0.75	0.79
multiBERT-MAX	0.80	0.76	0.81
multiBERT-LSTM	0.80	0.76	0.81
itaBERT-CLS	0.83	**0.80**	0.83
itaBERT-MAX	0.83	0.79	0.83
itaBERT-LSTM	**0.84**	0.79	**0.84**

Table: Emotion Detection Summary

Thank you!

Bag of Words

	bullismo	violenza	amore	donna	stupro
0	0	0	0	0	0
1	0	0	0	0	0
2	1	0	0	0	0
3	0	0	1	0	0
4	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮
710	0	0	2	0	0
711	0	0	1	0	0
712	0	0	0	0	0
713	0	1	0	1	0
714	0	0	0	0	0

Table: Bag of Words Representation

TF-IDF

	bullismo	**violenza**	**amore**	**donna**	**stupro**
0	0.00	0.00	0.00	0.00	0.00
1	0.00	0.00	0.00	0.00	0.00
2	0.19	0.00	0.00	0.00	0.00
3	0.00	0.00	0.20	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00
⋮	⋮	⋮	⋮	⋮	⋮
710	0.00	0.00	0.23	0.00	0.00
711	0.00	0.00	0.18	0.00	0.00
712	0.00	0.00	0.00	0.00	0.00
713	0.00	0.16	0.00	0.17	0.00
714	0.00	0.00	0.00	0.00	0.00

Table: TF-IDF Representation

TF-IDF Con't

$$\text{tf(w,D)} = \frac{f_{w,D}}{\sum_{w' \in D} f_{w',D}}$$

$$\text{idf(w)} = \log\left(\frac{1+N}{1+|\{D : w \in D\}|}\right) + 1$$

$$\text{TF-IDF(w,D)} = \text{tf(w,D)} \cdot \text{idf(w)}$$

◂ Data Preprocessing and Feature Engineering

Evaluation Metrics - Con't

$$\text{Accuracy} = \frac{\text{Total number of corrected classified tweets}}{\text{Total number of tweets}}$$

$$\text{Precision}_{\text{class A}} = \frac{\text{TP}_{\text{class A}}}{\text{TP}_{\text{class A}} + \text{FP}_{\text{class A}}}$$

$$\text{Recall}_{\text{class A}} = \frac{\text{TP}_{\text{class A}}}{\text{TP}_{\text{class A}} + \text{FN}_{\text{class A}}}$$

$$\text{F1-Score} = 2 \cdot \left(\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \right)$$

◂ Evaluation Metrics

VADER Example

Word	Compound Score	Standard Deviation	Raw Scores
lousiness	-1.7	0.64031	[-2, -2, -1, -1, -2, -2, -3, -2, -1, -1]
lousing	-1.1	0.9434	[-3, 0, 0, 0, -1, -2, -1, -2, -1, -1]
lousy	-2.5	0.67082	[-2, -4, -2, -3, -2, -3, -2, -2, -3, -2]
lovable	3.0	0.63246	[3, 3, 3, 4, 3, 2, 3, 3, 2, 4]
love	3.2	0.4	[3, 3, 3, 3, 3, 3, 3, 4, 4, 3]
loved	2.9	0.7	[3, 3, 4, 2, 2, 4, 3, 2, 3, 3]

Text	Negative	Neutral	Positive	Comp.
lotta bodyshaming essere cosa promuovere obesità essere altro	0.232	0.536	0.232	0.0
sperare solo cuore Giovanna avere capire parola bullismo uso sintetizzare	0.264	0.541	0.196	-0.25
seppellire odio sotto montagna amore ProudBoys	0.311	0.336	0.353	0.128

◂ Lexicon-based Approach: VADER

Support Vector Machines - Con't

Support Vector Machines optimization problem:

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\varepsilon_1,\ldots,\varepsilon_n,M}{\text{maximize}} \quad M$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1$$

$$c_i(\beta_0 + \beta_1 f_{i1} + \beta_2 f_{i2} + \ldots + \beta_p f_{ip}) \geq M(1 - \varepsilon_i) \quad \forall i = 1, \ldots, n.$$

$$\varepsilon_i \geq 0, \quad \sum_{i=1}^{n} \varepsilon_i \leq C$$

Radial basis function kernel:

$$g(f^*) = \beta_0 + \sum_{i \in S} \alpha_i K(f^*, f_i)$$

$$K(f_i, f_{i'}) = \exp\left(-\gamma \sum_{j=1}^{p} (f_{ij} - f_{i'j})^2\right)$$

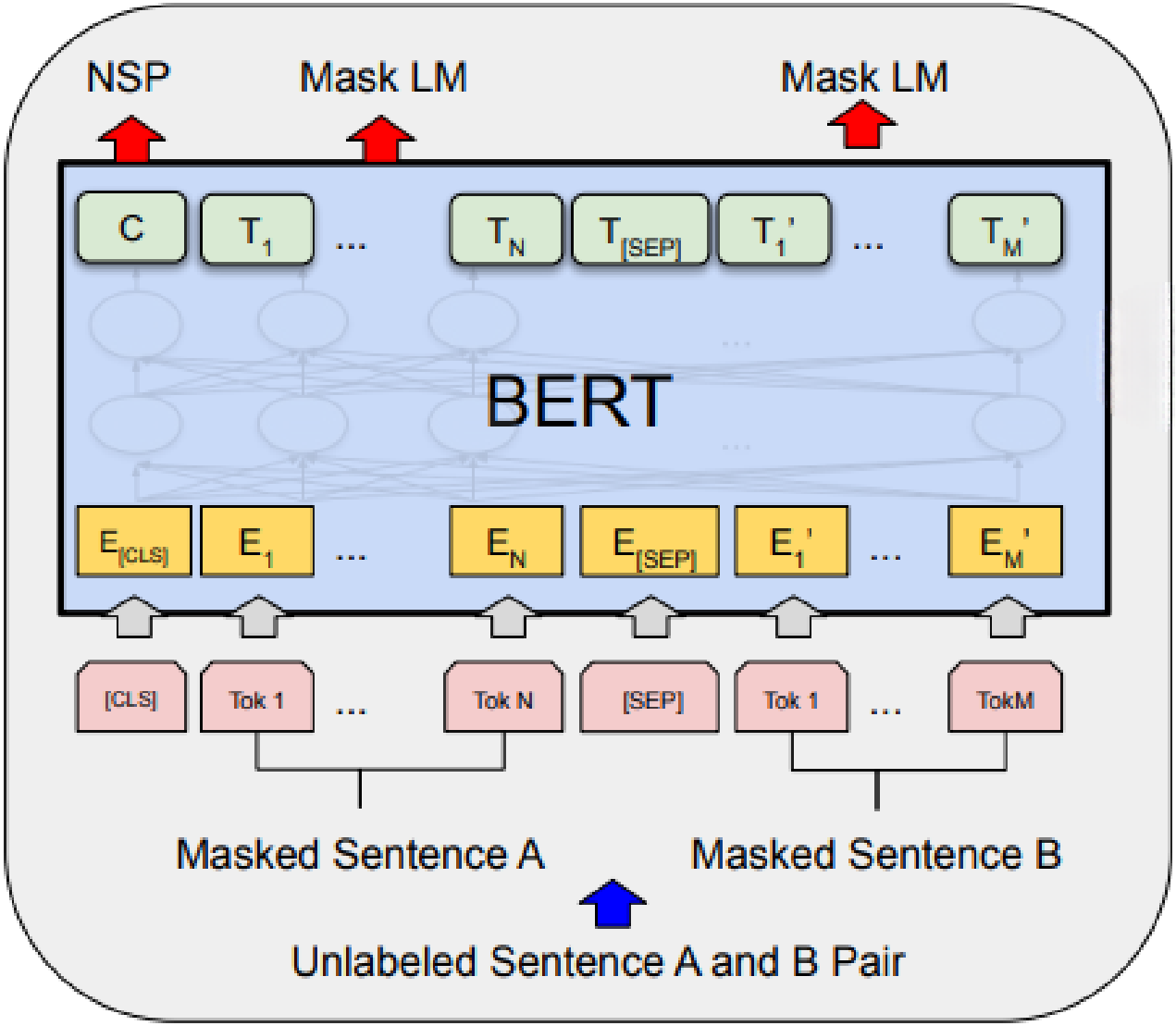
◂ Machine Learning Approach: Support Vector Machines

BERT

1. **Masked Language Modelling**: it consists of replacing 15% of the words in a given text sequence with a [MASK] token. The model's objective during pre-training is to predict the original words that have been masked, based on the context provided by the surrounding words in the sequence.

2. **Next Sentence Prediction**: the model is provided with pairs of sentences as input and is trained to predict whether the second sentence in each pair is indeed the subsequent sentence in the original document.

BERT Con't



Pre-training

WordPiece Algorithm

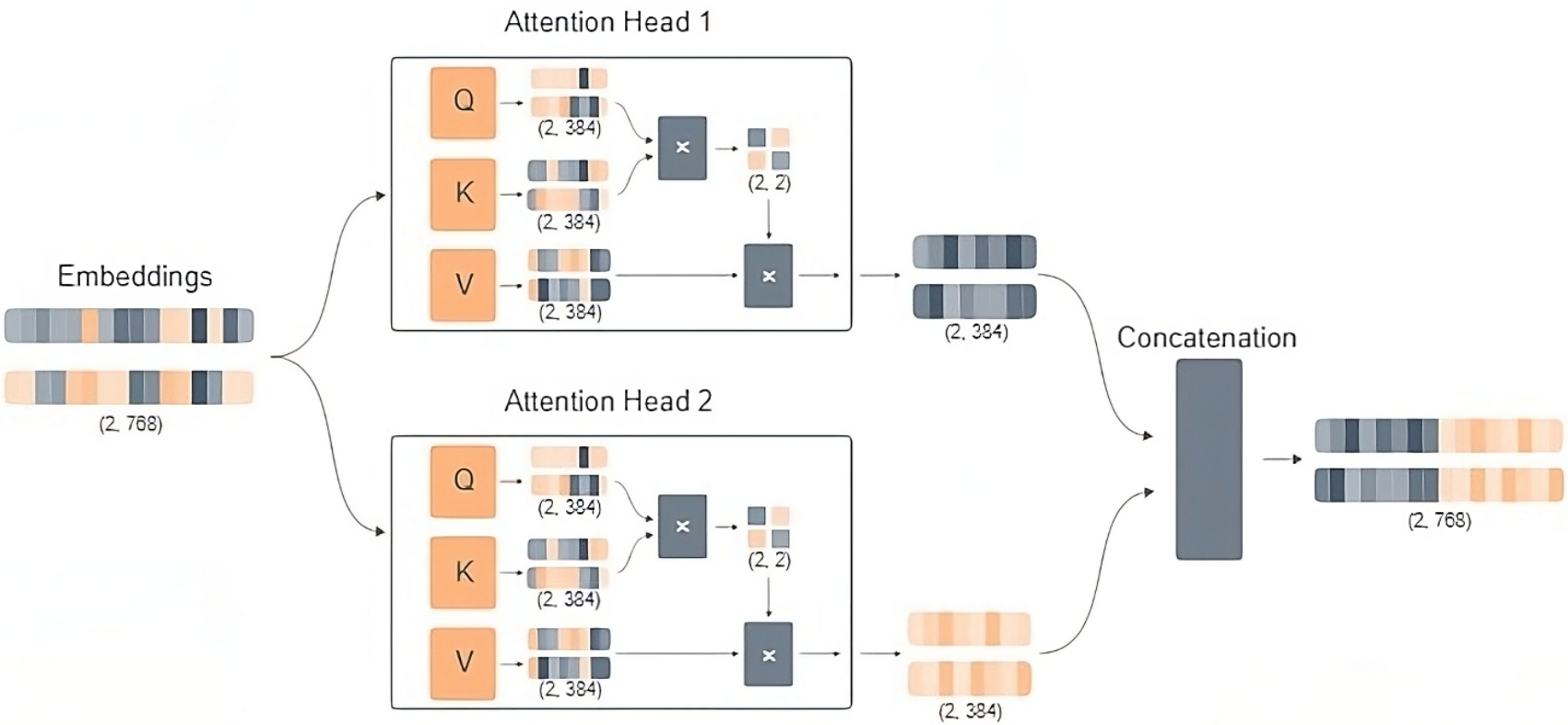
It starts with a small vocabulary including the special tokens and the alphabet. It identifies subwords by adding a prefix (##), so that each word is initially split by adding that prefix to all the characters inside the word.

Then, WordPiece algorithm learns merge rules to expand the vocabulary: it computes a score for each pair of subwords based on a weighted frequency formula.

$$\text{score} = \frac{\text{freq\_of\_pair}}{\text{freq\_of\_first\_element} \times \text{freq\_of\_second\_element}}$$

◂ BERT Encoding Pipeline: Tokenization

Attention Mechanism



◂ BERT Encoding Pipeline: Encoder Layer

BERT Model Variants

1. **bert-base-multilingual-cased**: The dataset used to train the model is wikipedia from Hugging Face, which contains 104 languages. For the italian language, the subset is composed by 1,743,035 training texts from a variety of subjects.

2. **dbmdz/bert-base-italian-xxl-cased**: Other than wikipedia, it has been trained on various texts from the OPUS corpora and data from the italian part of the OSCAR corpus, achieving a final training corpus of 81GB size and 13,138,379,147 tokens.

◂ Output Representations

Loss Function and Optimizer

- **Sparse Categorical CrossEntropy**

$$L_{CE} = -\sum_{c=1}^{M} y_c \ log(p_c)$$

- **AdamW**

Algorithm 2 Adam with L_2 regularization and Adam with decoupled weight decay (AdamW)

given $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, \lambda \in \mathbb{R}$
initialize time step $t \leftarrow 0$, parameter vector $\boldsymbol{\theta}_{t=0} \in \mathbb{R}^n$, first moment vector $\boldsymbol{m}_{t=0} \leftarrow \boldsymbol{0}$, second moment vector $\boldsymbol{v}_{t=0} \leftarrow \boldsymbol{0}$, schedule multiplier $\eta_{t=0} \in \mathbb{R}$
repeat
  $t \leftarrow t + 1$
  $\nabla f_t(\boldsymbol{\theta}_{t-1}) \leftarrow \text{SelectBatch}(\boldsymbol{\theta}_{t-1})$ ▷ select batch and return the corresponding gradient
  $\boldsymbol{g}_t \leftarrow \nabla f_t(\boldsymbol{\theta}_{t-1}) + \lambda \boldsymbol{\theta}_{t-1}$
  $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t$ ▷ here and below all operations are element-wise
  $\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1 - \beta_2)\boldsymbol{g}_t^2$
  $\hat{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t / (1 - \beta_1^t)$ ▷ β_1 is taken to the power of t
  $\hat{\boldsymbol{v}}_t \leftarrow \boldsymbol{v}_t / (1 - \beta_2^t)$ ▷ β_2 is taken to the power of t
  $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$ ▷ can be fixed, decay, or also be used for warm restarts
  $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t \left(\alpha \hat{\boldsymbol{m}}_t / (\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon) + \lambda \boldsymbol{\theta}_{t-1} \right)$
until *stopping criterion is met*
return optimized parameters $\boldsymbol{\theta}_t$

◂ Fine-Tuning