Italian Sentiment Analysis: Classifying Positive and Negative Words in Italian Articles

MATH 381 Final Project

Report by: Isaac Yeoh

Introduction

In this report, I aim to classify positive and negative Italian words through Italian articles by using the Least Squares Method, which is also known as Linear Regression. It is important to note that I possess no prior knowledge of the Italian language, ensuring that the project remains as unbiased as possible.

The Least Squares Method is an algorithm that computes a best fit line by minimizing and computing the following equation[1]:

$$R = \sum_{i=1}^{n} (y_i - f(x_i, a_i))^2 \rightarrow \min \tag{1}$$

In the equation above, a_i denotes the number of features, and $1 \leq i \leq n$ given n number of features. To minimize the least squares equation, we compute the partial differentiation of each feature. By equating the resulting partial derivatives to zero and solving the system of linear equations, we arrive at the values for the parameters that form the best fit line:

$$\Delta R = \begin{bmatrix} \frac{\partial R}{\partial a_1} \\ \vdots \\ \frac{\partial R}{\partial a_n} \end{bmatrix} = 0 \tag{2}$$

In addition to the Linear Regression model, I will also compare the results obtained from this model with other models to observe how different models classify terms differently.

Data Collection

To conduct sentiment classification on Italian words, I sourced various Italian articles and assigned a subjective score to them. A large majority of my articles were sourced from the online Italian article website la repubblica. The other articles were sourced from tabloids and websites that allowed a limited number of views before locking the articles

behind a paywall. In each article, I first translated the title using DeepL, a machine learning translation model that has a fairly good reputation for its translation accuracy.

Next, I copied the contents of the articles, which are in Italian and are not translated, into a spreadsheet. I then assigned them a score based on the translated article title that ranged from 1 to 5. In this scale, 1 denoted the most negative sentiment, 5 denoted the most positive sentiment, and 3 represented a neutral stance. In instances where articles contained a mix of both positive and negative statements within a sentence, I also assigned a score of 3. One such example of that is: ”Lottery, wins 10 million in Germany and girlfriend blows her nose with a 500-euro bill: video sparks a storm”. Since winning the lottery is a positive statement and ”sparks a storm” is a negative statement, it was designated as neutral. Figure 1 below showcases a sample of my data:

	A	B	C	D
1	translated_title	score	text	source
2	Sinner-Rune, the Atp Finals	4	Ore 00:05 - Il terzo capolavoro di Sinner(Gai	https://www.corri
3	Giulia Cecchettin and Filipp	1	Treviso, l'immagine risale alla notte tra sabat	https://corrierede
4	Israel - Hamas at war, toda	1	È il 41esimo giorno di guerra: il bilancio tra i	https://www.corri

Figure 1: Sample Data

Pre-processing

For each article, I mapped the scores ranging between 1 to 5 to a new sentiment score ranging between -2 to 2:

$$1 \rightarrow -2$$

$$2 \rightarrow -1$$

$$3 \rightarrow 0$$

$$4 \rightarrow 1$$

$$5 \rightarrow 2$$

Subsequently, I excluded ineligible texts and numerals from the set of article con-

tents. Numerals had to be removed as the dates and time were copied along with the article contents. Note that numbers written in text form were not removed. I also contemplated removing punctuation from the set of articles, but I observed a significant usage of apostrophes and concluded that retaining the punctuation would enhance the contextualization of the language when running the model. After doing so, I had to introduce a set of Italian stop words to filter out the commonly used words as seen in Figure 2 below:

```
italian_stop_words = ["del", "suo", "con", "più", "di", "mi", "oggi",
                      "novembre", "le", "da", "fare", "il", "come", "dei",
                      "se", "ho", "la", "totti", "alla", "dalle", "sarà",
                      "diventato", "tutto"]
```

Figure 2: Italian Stop Words

The final pre-processing step that was done was to create a vector containing the tf-idf value of the words. Tf-idf is an abbreviation for Term Frequency - Inverse Document Frequency. It measures how important a term is within a document relative to the collection of document. The tf-idf values are calculated using the formulas denoted below[2]:

Term Frequency (tf):

$$\text{tf}(t,d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d} \tag{3}$$

Inverse Document Frequency (idf):

$$\text{idf}(t,D) = \log\left(\frac{\text{Total number of documents in the corpus } N}{\text{Number of documents containing term } t + 1}\right) \tag{4}$$

Combining them, we obtain $0 \leq \text{tf-idf} \leq 1$:

$$\text{tf-idf} = \text{tf}(t,d) \times \text{idf}(t,D) \tag{5}$$

After obtaining the tf-idf values for each term, I filtered out the most common terms and least common terms to avoid potential misclassification in the model by adjusting the

conditions of tf-idf to $0.05 \leq$ tf-idf ≤ 0.85 instead. The sklearn package was used to compute the tf-idf values in my code, which could be seen in Figure 3 below:

```
X = articles['text_clean']
y = articles['sentiment']

italian_stop_words = ["del", "suo", "con", "più", "di", "mi", "oggi",
                      "novembre", "le", "da", "fare", "il", "come", "dei",
                      "se", "ho", "la", "totti", "alla", "dalle", "sarà",
                      "diventato", "tutto"]
stop_words = list(ENGLISH_STOP_WORDS) + italian_stop_words

vectorizer = TfidfVectorizer(max_features=100000, stop_words=stop_words,
                             max_df=0.85, min_df=0.05)
X_tfidf = vectorizer.fit_transform(X)
```

Figure 3: Creating the tf-idf vector after filtering out stop words

Creating the Linear Regression Model

Using sklearn's Linear Regression package, I was able to create the model by setting the model's feature values as the tf-idf values and setting the target values as the sentiments. The code used to create the model is seen in Figure 4 below:

```
X_train, X_test, y_train, y_test = train_test_split(X_tfidf,
                                                    y,
                                                    test_size=0.2,
                                                    random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)

coefficients = model.coef_
top_negative_indices = np.argsort(coefficients)[:20]
top_positive_indices = np.argsort(coefficients)[-20:][::-1]
features = vectorizer.get_feature_names_out()
top_negative_words = [features[i] for i in top_negative_indices]
top_positive_words = [features[i] for i in top_positive_indices]

print("Top 20 Negative Words:", top_negative_words)
print("Top 20 Positive Words:", top_positive_words)

y_pred = model.predict(X_test)
print(model.predict(X_test))
print(y_test)

print("Mean squared error:", mean_squared_error(y_test, y_pred))
```

Figure 4: Linear Regression Model Code

Listed below is a list of the top 20 classified words of each category by the model.

Top 20 **Negative** Words, ranked from most negative to least negative:

1. 'ragazzi' (guys)
2. 'madre' (mother)
3. 'morta' (dead)
4. 'tre' (three)

5. ’detto’ (said)
6. ’testa’ (head)
7. ’artificiale’ (artificial)
8. ’presidente’ (president)
9. ’figlia’ (daughter)
10. ’trovato’ (found)
11. ’foto’ (photo)
12. ’quattro’ (four)
13. ’autorità’ (authority)
14. ’denuncia’ (complaint)
15. ’gatto’ (cat)
16. ’punto’ (point)
17. ’persone’ (people)
18. ’modo’ (way)
19. ’ragazza’ (girl)
20. ’ormai’ (by now)

Top 20 **Positive** Words, ranked from most positive to least positive:

1. ’nazionale’ (national)
2. ’grazie’ (thank you)
3. ’comitato’ (committee)
4. ’si’ (yes)

5. ’uno’ (one)
6. ’città’ (city)
7. ’pace’ (peace)
8. ’quando’ (when)
9. ’infatti’ (indeed)
10. ’storia’ (story)
11. ’nuova’ (new)
12. ’stava’ (was)
13. ’ue’ (EU)
14. ’opere’ (works)
15. ’squadra’ (team)
16. ’sul’ (on the)
17. ’subito’ (immediately)
18. ’sindaco’ (mayor)
19. ’premio’ (prize)
20. ’ogni’ (every)

Upon initial inspection of the top 20 words in each category, the positive classification of the words appear to be relatively accurate, however, the negative classification contained many neutral words such as ”guys”, ”mother”, ”three”, and ”daughter”. To validate the accuracy of the model, I conducted a manual inspection of the top 20 positive and negative words in the article dataset. The classification of words consistently aligns with the scores provided in the dataset, which suggests that the model is working as intended. If you are perplexed about the inclusion of words such as ”cat”, ”four”, ”three” and ”daughter” in the

negative classification, it is primarily due to their association with articles discussing topics related to murders, kidnappings, and abuse.

Comparing Other Models

In addition to the Linear Regression Model, I created a Logistic Regression Model using the same pre-processing steps. The logistic regression model uses a sigmoid logistic function to transform the output of a linear equation into a probability score between 0 and 1. The sigmoid function is defined as[3]:

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{6}$$

z is denoted as the linear combination of x_i, the TF-IDF values for each word, and β_i, the coefficients corresponding to the respective independent variables, where $0 \leq i \leq n$ given by the formula:

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n \tag{7}$$

The logistic regression model was executed with regularization, which is used to prevent overfitting. It is determined by the λ value which controls the degree of regularization. The lambda value was determined through cross-validation, which in this case is $\lambda = 0.001$. The results for the top positive and negative sentiments are listed below:

Top 20 **Negative** Words, ranked from most negative to least negative:

1. 'uccisione' (killing)
2. 'morta' (dead)
3. 'gaza' (Gaza)
4. 'trovato' (found)
5. 'testa' (head)

6. ’turetta’ (turetta)
7. ’intelligenza’ (intelligence)
8. ’polizia’ (police)
9. ’netanyahu’ (Netanyahu)
10. ’hamas’ (Hamas)
11. ’cimitero’ (cemetery)
12. ’vienna’ (Vienna)
13. ’filippo’ (Filippo)
14. ’cani’ (dogs)
15. ’stevens’ (Stevens)
16. ’sparato’ (shot)
17. ’heather’ (Heather)
18. ’capretta’ (goat)
19. ’gatto’ (cat)
20. ’uomo’ (man)

Top 20 **Positive** Words, ranked from most negative to least negative:

1. ’maggior’ (major)
2. ’moda’ (fashion)
3. ’rosa’ (pink)
4. ’nobel’ (Nobel)
5. ’mohammadi’ (Mohammadi)

6. ’francesco’ (Francesco)
7. ’lavoro’ (work)
8. ’mare’ (sea)
9. ’parte’ (part)
10. ’anno’ (year)
11. ’ue’ (EU)
12. ’storia’ (history)
13. ’europa’ (Europe)
14. ’lotto’ (lottery)
15. ’città’ (city)
16. ’nazionale’ (national)
17. ’infatti’ (indeed)
18. ’contro’ (against)
19. ’tv’ (TV)
20. ’italia’ (Italy)

Upon analysis, I noticed that this model was much better at classifying negative sentiments. However, the positive sentiments were not classified as well as the Linear Regression Model. Furthermore, after manual comparison of the words classified as positive by the model with the scores assigned in my dataset, I observed that the scores did not accurately reflect the classification made by the model. For example: ”maggior” had an average score of 3.3, ”moda” had an average score of 2.75, and ”contro” had an average score of 2.82 in the dataset.

I also developed a Linear Regression Model that doesn't preprocess the tf-idf values to observe the differences. To prevent any potential confusion between this new model and the previous Linear Regression Model, I'll refer to this model as the 'no-tfidf' model.

The no-tfidf model classified words very similarly to the Linear Regression model. However, one notable difference between these two models was that the no-tfidf model had significantly more neutral words mixed into the classified words. A snippet of that can be seen in Figure 5 below:

```
Top 5 Positive Words: ['si', 'uno', 'proposta', 'nazionale', 'sua', 'all', 'quando', 'mohammadi', 'lo
Top 5 Negative Words: ['austria', 'che', 'sparato', 'non', 'vienna', 'cimitero', 'nel', 'lavoratori',
```

Figure 5: no-tfidf Top Positive and Negative Words

Upon reviewing the outcomes, it became apparent that the word classifiers contained more countries and names, which is what we want to avoid. Hence, the tf-idf pre-processing step is necessary to remove a large majority of unwanted terms in the final result.

Conclusion

Overall, the Linear Regression model managed to capture the sentiments of the words relatively accurately. Although some neutral words were included in both the positive and negative classifiers, a manual inspection of the scores in the dataset reveals that the predicted words consistently align with the average scores among articles containing those words.

To improve the Linear Regression Model, the dataset size could be increased. Currently, the dataset comprises 110 articles. With more articles, the frequency of words would also make the model more accurate in classifying positive and negative sentiments.

Citations

1) “Least Squares Fitting.” 2023. Wolfram.com. Wolfram Research, Inc. 2023. https://mathworld.wolfram.com/LeastSquaresFitting.html.

2) “Understanding TF IDF Term Frequency Inverse Document Frequency.” 2021. GeeksforGeeks. GeeksforGeeks. January 20, 2021. https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/.

3) “Educative Answers - Trusted Answers to Developer Questions.” 2015. Educative. 2015. https://www.educative.io/answers/what-is-sigmoid-and-its-role-in-logistic-regression.