Text Analytics
EXAMINATION PROJECT REPORT

Figurative Language Detection in Tweets:
a Comprehensive Analysis for Exploring Irony and Sarcasm through Semantic Enrichment

Jacopo Gneri (581536)[1]
Valeria Picchianti (562913)[1]
Andrea Frasson (657944)[1]
Vincenzo Sammartino (599203)[2]
Corrado Baccheschi (599107)[2]

[1] *Master's degree in Data Science and Business Informatics, University of Pisa*
[2] *Master's degree in Digital Humanities, University of Pisa*

ACADEMIC YEAR 2023-2024

Contents

1 Introduction

The rapid and concise nature of communication on social media platforms has made it imperative to grasp the nuances of figurative language and irony in online messages. Accurate interpretation is vital to prevent misunderstandings in this fast-paced digital landscape. This project aims to enhance our understanding of these linguistic subtleties, offering the potential to improve the interpretation of online content and address challenges related to digital communication.

This project was sparked by the inspiration drawn from the work of Recupero et al. (2019), as presented in their paper titled *Frame-Based Detection of Figurative Language in Tweets* [3]. However, it also aims to build upon their results by enriching, expanding, and further improving the understanding of figurative language and irony in social media communication.

Our primary objective is to identify figurative language in tweets and, where present, detect sarcasm and irony. We also plan to compare the performance of several classification models trained on both the original dataset used in the reference paper and an extended and cleaned dataset augmented with additional features containing knowledge provided by SenticNet. In this report we will outline our methodology, including the implementation of models and their evaluation.
The key steps followed to accomplish the goals may be summarised as:

1. Data Exploration and Preprocessing: we begin by thoroughly understanding the dataset and preparing it for further analyses. This step involves also enriching the dataset with information from SenticNet [1] and performing data cleaning and feature selection to remove inconsistencies and ensure data quality;
2. Classification: This key step involves the implementation of classification models to identify figurative language, and subsequently also sarcasm vs irony;
3. Analysis of the Results: We analyze the results obtained from the classification models, also comparing such results with those achieved in the reference paper;
4. Explainability: In the pursuit of interpretability, we focus on explaining the outcomes and the decision-making process of our models.

The datasets used in this project are drawn from *SemEval-2015 Task 10: Sentiment Analysis in Twitter* [2] and *SemEval-2015 Task 11: Sentiment Analysis of Figurative Language in Twitter*[3].

2 Background

In this project various machine learning and deep learning models were employed for the classification task, aimed at identifying figurative language in tweets. Naive Bayes, Support Vector Machines (SVM), Decision Trees, k-Nearest Neighbors (KNN) and Random Forest are traditional machine learning algorithms that operate on different principles. Naive Bayes relies on probabilistic inference, assigning probabilities to different classes based on the occurrence of features. SVM aims to find an optimal hyperplane that separates classes in a high-dimensional space. Decision Trees recursively split data based on features to form a tree structure for classification. KNN classifies instances based on the majority class of their k-nearest neighbors in the feature space. Random Forest builds an ensemble of decision trees for improved accuracy and robustness. Moreover, deep learning models such as BERT, RoBERTa, and LSTM delve into the semantic complexities of language. BERT and RoBERTa use transformer architectures to capture contextual information, while LSTM (Long Short-Term Memory) is a recurrent neural network that excels in handling sequential data.
The comparisons between algorithms provide valuable insights into the operating principles and assumptions defined by each model. Understanding the differences and exploiting how the algorithms behave in the same scenario is the foundation for future and more complex studies.

[1]https://sentic.net/
[2]https://alt.qcri.org/semeval2015/task10/
[3]https://alt.qcri.org/semeval2015/task11/

3 Methodologies

This section delineates the comprehensive methodologies employed for the project, encompassing the preprocessing step, classification task and the subsequent model explainability analysis.

3.1 Data Exploration and Preprocessing

The preprocessing pipeline is represented in Figure 1.

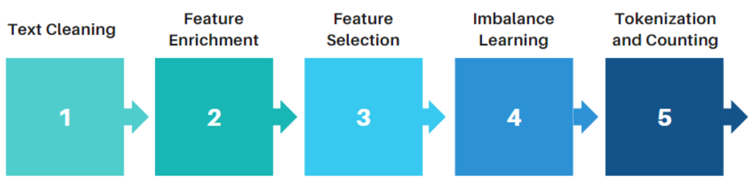


Figure 1: Pipeline of the preprocessing task

3.1.1 Text cleaning

The text cleaning process involves a series of steps to enhance the quality and uniformity of the textual data. First, the text in each tweet is converted to lowercase. This process employs regular expressions to remove mentions (prefixed with '@') and hyperlinks (starting with 'http://' or 'https://') from the text. Furthermore, the function targets specific URL patterns related to Instagram, Pinterest images and general web links for their removal. Additionally, a regular expression is applied to eliminate punctuation, while consecutive double spaces are replaced with a single space. Stop words are removed from the cleaned text. A list of them is obtained using the `stopwords.words('english')` function from the NLTK library. Emoticons are identified and extracted using a specific pattern, and so are the hashtags: we decided not to remove them from the text during the cleaning process, but instead, their presence or absence is recorded as a binary indicator. This choice stems from the belief that hashtags and emoticons can carry discriminative information for the classification of figurative or non-figurative text.

3.1.2 Feature Enrichment

Syno_Lower_Mean* and *Syn_Mean In this project we introduced two novel features to enhance the analysis of ironic intent in tweets [2]. Leveraging WordNet, a lexical database, we incorporated the *Syno_Lower_Mean* feature, calculated as the mean frequency of synonyms with lower frequencies than the original word in a given tweet, excluding stop words. Additionally, the *Syn_Mean* feature represents the mean frequency of synonyms for all words in the tweet. These features aim to capture the nuanced choice of words in ironic communication, where authors strategically select less common terms to convey both a literal and a figurative message simultaneously. It follows a detailed description of each of the two:

1. *Syno_Lower_Mean*: For each word in a tweet (after removing stopwords), its synonyms are retrieved using WordNet. Using the ANC (American National Corpus[4]) frequencies, the frequency of each synonym are computed and filtered keeping the ones with frequencies lower than the frequency of the original word. *Syno_Lower_Mean* feature is computed as the mean frequency of these selected synonyms for each word in the tweet. This feature aims to quantify the choice of less common synonyms in ironic communication, capturing the deliberate use of words that diverge from more conventional options.

2. *Syn_Mean*: Similar to the previous one, synonyms for each word in the tweet using WordNet are retrieved and ANC frequencies are used to calculate the frequency of each synonym. *Syn_Mean* feature is the mean frequency of all synonyms for each word in the tweet. This feature provides a broader perspective by considering all synonyms, regardless of their frequency relative to the original word.

[4]`https://anc.org/`

SenticNet In the process of enhancing tweets, SenticNet (a semantic resource) was employed to extract sentiment-related information. Each word in a tweet underwent analysis using SenticNet, yielding data encompassing polarity labels, polarity values, moodtags, semantics, and sentics (semantic dimensions with associated values). Basically, it iterates through the dataset, applying the SenticNet analysis to each tweet and organizing the outcomes into a new and enriched DataFrame: it integrates columns related to sentiment, such as *positive* and *negative*, along with individual columns for each unique moodtag and semantic. Additionally, it separates sentic dimensions into distinct columns. The result is a more nuanced representation of emotional and semantic nuances in each tweet, enabling a comprehensive investigation into the sentiments and meanings expressed in natural language.

Framester The reference dataset was already enriched with information coming from Framester[5], a wide coverage hub of linguistic linked data standardized in the form of a knowledge graph based on Frame Semantics. Framester facilitated the extraction of semantic features from the text, incorporating event information using frames, word senses, and lexical units. This augmentation enabled a more robust exploration of sentiment and meaning in natural language. An example of annotations with Framester is shown in Figure 2.

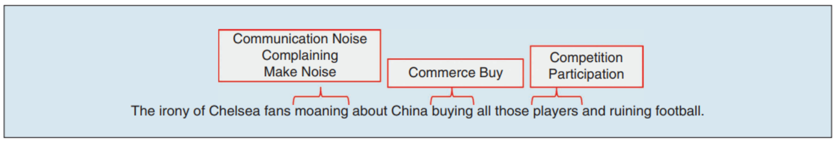

Communication Noise
Complaining
Make Noise
Commerce Buy
Competition
Participation
The irony of Chelsea fans moaning about China buying all those players and ruining football.

Figure 2: Example of sentence annotated with Framester

3.1.3 Feature Selection

Finally, the decision tree-based feature selection technique was applied: 200 trees were trained using random parameters to gain insight on each variable's importance. This approach facilitated the exclusion of less significant variables and retained only those features which were selected more times by models, that is the features whose importance was above 0.1%. The primary motivation for adopting this strategy was the intention to retain features (words) whose impact on the overall predictive performance was relatively substantial. The dataset resulting from this selection consists of 49 variables.

3.1.4 Imbalanced Learning

The original dataset encompassed a target class (score of each tweet) with values ranging from -4 to +4. To reduce it to a binary classification problem, we transformed the feature distinguishing between figurative and non-figurative text. This conversion involved mapping the original values to a binary format, where values greater or equal than 0 were mapped to 0 (indicating non-figurative text) and values smaller than 0 were mapped to 1 (indicating figurative text). Upon examining the original dataset, we observed a significant class imbalance in the target variable, with counts: 421 instances for Class 0 vs 5889 instances for Class 1. Given this substantial imbalance, where Class 1 significantly outnumbered Class 0, undersampling was implemented to address such issue, ensuring a more equitable representation of minority and majority classes in the dataset. The random undersampling technique involved randomly selecting a subset of instances from the majority class to create a more balanced training set. This approach aimed to enhance the model's ability to learn from both classes, ultimately improving its predictive performance on the minority class, avoiding the risk of including biases due to oversampling.

3.1.5 Tokenization and counting

CountVectorizer was harnessed to create a traditional bag-of-words representation, quantifying the frequency of individual terms across the entire corpus. Additionally, Word2Vec was applied to produce dense vector embeddings, capturing semantic relationships and contextual nuances within the text. This dual-processing strategy yielded two separate datasets, each with its own set of features and representations.

[5]`https://framester.github.io/`

3.2 Classification

To optimize the performance of the selected models, a systematic approach to hyperparameter tuning was adopted. Grid search and randomized search techniques were employed to explore and identify the most effective hyperparameter configurations for each model. This process aimed to enhance the models' predictive capabilities and generalization to unseen data. The model building process involved the use of differently preprocessed datasets, including those pre and post feature selection, and incorporating various encoding techniques. This approach allowed for a comprehensive exploration of the impact of data preprocessing on model performance. It ensured that the models were trained on well-processed and representative data, thereby improving their ability to capture the nuances of figurative language in tweets. To assess the models' performance, a comprehensive evaluation strategy was employed. Metrics such as accuracy, precision, recall, and F1-score were computed. Furthermore, confusion matrices were analyzed to gain insights into the models' ability to correctly classify figurative language instances.

3.3 Explainability

The methodologies employed to enhance the interpretability and explainability of the models are presented: we primarily focused on decision trees, their feature importance analysis and the feature importance analysis of Support Vector Machines (SVM). Decision trees inherently provide a transparent and interpretable representation of the decision-making process within the model. We leveraged decision trees to gain insights into the hierarchical structure of features influencing the classification of figurative language in tweets. To delve deeper into the significance of individual features, we conducted an analysis of feature importance within decision trees and also SVM. This involved assessing the contribution of each feature to the overall predictive performance of the model. By quantifying feature importance, we identified linguistic and contextual key elements that played pivotal roles in the model's ability to discern figurative language.

4 Case Study and Results

4.1 Choice of Dataset and Data Preprocessing

The selection of the dataset for this project aligns with the reference paper, providing a foundation for meaningful comparisons between our results and those of the original study. The dataset employed for experimentation comprises tweets sourced from diverse corpora, encompassing both figurative language (containing irony and sarcasm) and non-figurative expressions. To address the tasks of detecting figurative language and classifying tweets with irony and sarcasm, tweets from SemEval-2015 Task 10 and Task 11 were used. The comprehensive dataset incorporated both figurative and non-figurative tweets. To enhance the input datasets, we extended them by including additional features as explained in section 3.1.2.
The dataset is then preprocessed as outlined in section 3.1 following these steps: text cleaning, feature enrichment with *Syno_Lower_Mean* and *Syn_Mean*, feature selection based on decision trees whose selection consisted of 49 variables, imbalanced learning and vectorization. Various preprocessed datasets are employed for model training, encompassing datasets pre and post feature selection, along with distinct encoding techniques.

4.2 Classification 1: *Figurative* vs *Non Figurative language*

To demonstrate the importance and effectiveness of the preprocessing step, the classification models were initially trained and tested on the raw dataset: in all cases, this essentially highlighted the limitations of the classifiers, as the models struggled to achieve accurate or consistent results due to the presence of noise, unstructured data or redundant information in the raw dataset. Subsequently, by applying the preprocessing described above (section 3.1), a significant improvement in the performance of the classifiers was observed. This evidence clearly emphasizes how crucial it is to implement a well-designed preprocessing step to ensure that classification models can learn effectively and produce reliable results.

While acknowledging the significance of the preprocessing step, it is important to emphasize that, for certain models, the reduction in the number of features resulting from the feature selection process may not always be advantageous.

4.2.1 Support Vector Machines

In the case of classification with Support Vector Machines, our focus was on analyzing three scenarios:

a) Training the model without incorporating class weights.

b) Training the model with class weights, specifically {0: 0.85, 1: 0.7}.

c) Training the model with class weights, {0: 0.35, 1: 0.4}, utilizing only two features, namely *Syn_Lower_Mean* and *Syno_mean*. It is worth noting that these two features were introduced during the preprocessing step (Section 3.1).

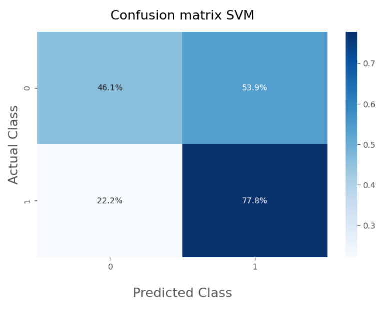
For all the three models, a thorough hyperparameter tuning process was undertaken.

A detailed analysis of the confusion matrices for three classification scenarios (Figure 3) is presented:

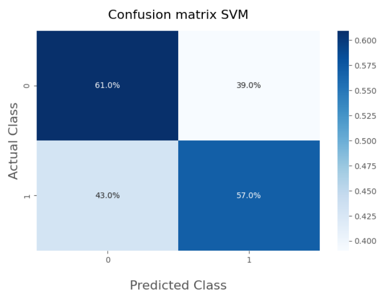
Case (a) - Without Class Weights: Precision: 0.67, Recall: 0.46, F1-Score: 0.55. Higher precision for Class 1 suggests a tendency to correctly identify positive instances. Overall model accuracy is moderate at 0.62.

Case (b) - With Class Weights: Precision: 0.58, Recall: 0.61, F1-Score: 0.60. Slight decreases in precision and recall compared to Case (a), indicating a trade-off. The model aims to balance misclassifications, resulting in an accuracy of 0.59.

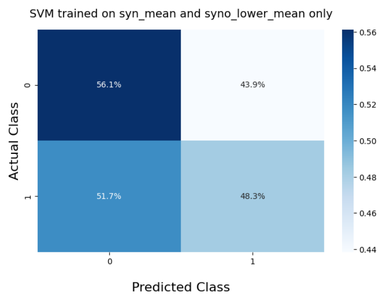
Case (c) - With Class Weights, 2 Features Only: Precision: 0.52, Recall: 0.56, F1-Score: 0.54. Lower precision and recall values suggest potential information loss with feature selection. However, it is noticeable that even with only 2 features, the model's performance remains decent, showing their importance.



(a) Without class weights



(b) With class weights



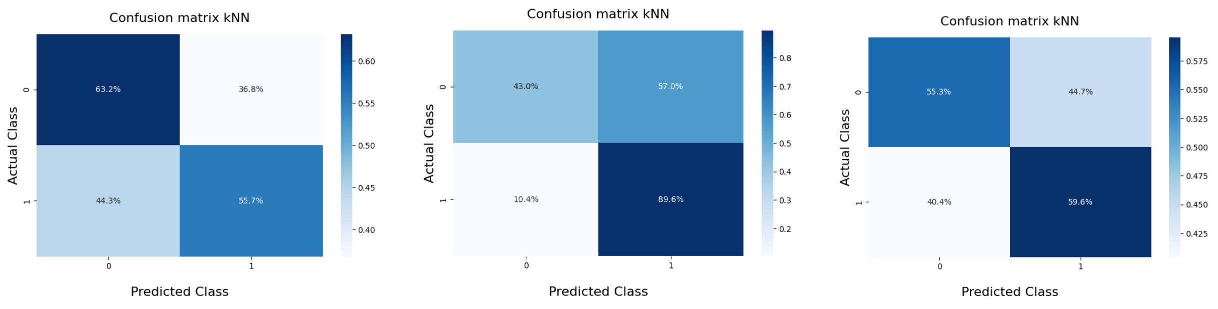
(c) With class weights - 2 features only

Figure 3: Comparing performance of three different SVC models: (a) Without class weights, (b) With class weights, and (c) With class weights and focusing on 2 features only, that is *Syn_Lower_Mean* and *Syno_mean*.

4.2.2 k-Nearest Neighbors

In this section we explore the performance of the kNN model trained on the 'full' dataset, the reduced one and the one with SenticNet information.

The model trained on the dataset without feature selection demonstrates moderate performance, achieving an accuracy of 0.59. Introducing SenticNet information to the dataset significantly improves the model's accuracy to 0.66, with notable enhancements in precision and recall for both classes. Feature selection on the dataset, however, results in a slight decrease in accuracy to 0.57, suggesting that, in this case, the reduction of features did not contribute positively to model performance.



(a) Without feature selection (b) With SenticNet info (c) With feature selection

Figure 4:

4.2.3 Logistic Regression

Since the target variable is binary, we found it interesting to also test the classification using logistic regression (LR). When using either CountVect or embeddings on the 'full' dataset (without feature selection and without SenticNet information), the model performs poorly, essentially yielding random results:

	Precision	Recall	F1-Score	Support
Class 0	0.50	1.00	0.66	228
Class 1	0.00	0.00	0.00	230
Accuracy			0.50	458
Macro Avg	0.25	0.50	0.33	458
Weighted Avg	0.25	0.50	0.33	458

Table 1: Logistic Regression Classification Report - base scenario

In order to enhance our results, we opted to explore techniques for handling negation. These techniques are designed to improve the model's capability to recognize shifts in sentiment influenced by the presence of negation words. The addition of '_NEG' to the feature set arises from the implementation of negation handling on the text data (`nltk.sentiment.util.mark_negation` function[6]). This process entails annotating words within a negation scope, indicating a potential alteration in sentiment. An example of our usage for the function:

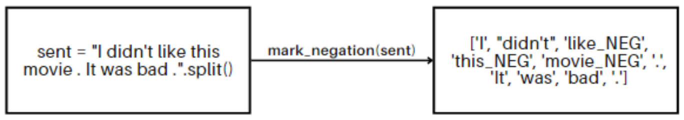

sent = "I didn't like this movie . It was bad .".split()
mark_negation(sent)
['I', "didn't", 'like_NEG', 'this_NEG', 'movie_NEG', '.', 'It', 'was', 'bad', '.']

Figure 5: An example of `nltk.sentiment.util.mark_negation`, appending '_NEG' to words that appear in the scope between a negation and a punctuation mark

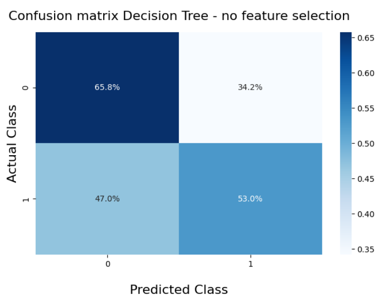
Including _NEG features with CountVect or embeddings improves the model's performance. Logistic regression exhibits varying performance based on the inclusion or exclusion of _NEG features, emphasizing the need for careful consideration of feature selection to achieve optimal results. In this case the classification report indicates that the model achieved balanced precision and recall for both classes (0 and 1), resulting in an overall accuracy of 65%, demonstrating a more well-performing and consistent classification across the dataset.

4.2.4 Decision Tree Classifier

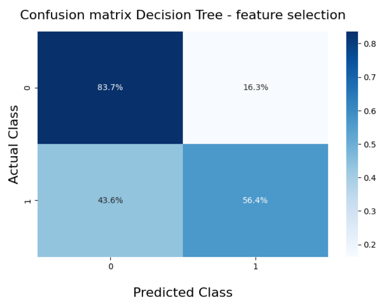
Also the Decision Tree Classifier was evaluated in various scenarios, considering different feature sets and embeddings. Figure 6 displays the confusion matrices for the decision tree. The tree was trained on the

[6]https://www.nltk.org/api/nltk.sentiment.util.html

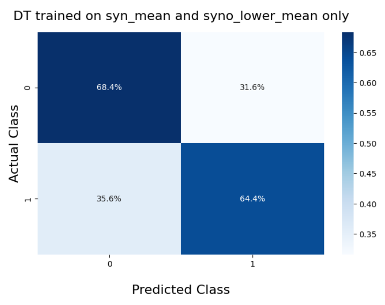
entire original feature space, on the selected dataset, and lastly, using solely the two features introduced by us, namely *Syn_Lower_Mean* and *Syno_mean*. The plots reveal an enhancement between case a) and case b) in the model's capability to differentiate class 0 without compromising performance on class 1. It is noteworthy that in case c), which utilizes only two features, the model maintains comparable metrics (slightly improved) as in case a), but with a considerably reduced computational load.



(a) Without feature selection



(b) With feature selection



(c) With 2 features only

Figure 6: Comparing performance of three different DT models: (a) Without feature selection, (b) With feature selection, and (c) With 2 features only, that is *Syn_Lower_Mean* and *Syno_mean*.

We opted to evaluate how the use of distinct encoding techniques, specifically Word2Vec and Doc2Vec, would affect the classification results in comparison to the previously mentioned case (with CountVectorize). Accuracy values on test set: 58,52% with Doc2Vec and 49,12% with Word2Vec.

4.2.5 Random Forest

During our investigation of the Random Forest model, the training process revealed a performance decline with the introduction of _NEG features and a certain sensitivity to encoding techniques, including CountVectorizer and Word2Vec. More precisely, the comparison between classification reports is outlined below (Figure 7). This highlights a noteworthy impact on the model's performance associated with the inclusion of _NEG features. In general, the Random Forest model appears to be less robust when these features are considered, and it demonstrates relatively better performance without them.

	Precision	Recall
Class 0	0.68	0.53
Class 1	0.62	0.76
Accuracy		0.64
Macro Avg	0.65	0.64
Weighted Avg	0.65	0.64

(a) With word embeddings and without _NEG features

	Precision	Recall
Class 0	0.61	0.57
Class 1	0.60	0.63
Accuracy		0.60
Macro Avg	0.60	0.60
Weighted Avg	0.60	0.60

(b) With word embeddings and _NEG features

Figure 7: Comparison of performances of Random Forest trained on different preprocessed datasets

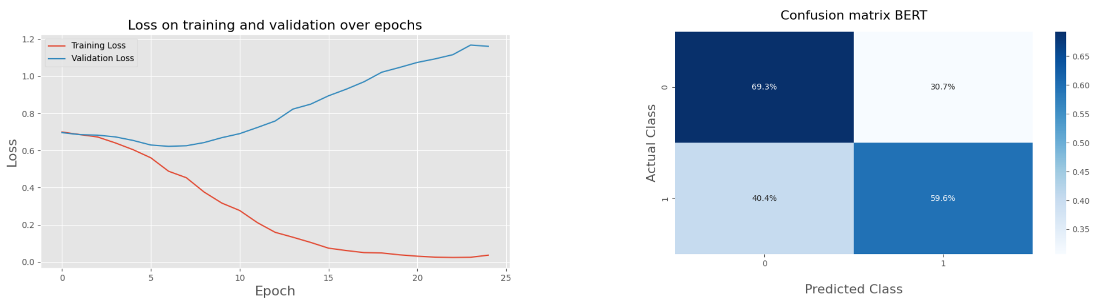
Table 2 represents the performance metrics of the best-trained model to date, employing the Random Forest algorithm with CountVectorizer and without _NEG features. With a balanced combination of precision, recall, and F1-score, this model achieved an accuracy of 70%, making it the most successful among the ones considered.

	Precision	Recall	F1-Score	Support
Class 0	0.74	0.60	0.66	228
Class 1	0.67	0.79	0.72	230
Accuracy			0.70	458
Macro Avg	0.70	0.70	0.69	458
Weighted Avg	0.70	0.70	0.69	458

Table 2: Classification Report of the best trained model: Random Forest with CountVectorizer, without _NEG features

4.2.6 BERT

We also opted to explore BERT for its enhanced performance attributed to contextualized embeddings, pre-trained representations, and transfer learning capabilities. However, looking at Figure 8 in the initial epochs, training loss steadily diminishes, signifying positive learning from the training data. Simultaneously, the validation loss decreases, indicating effective generalization to unseen data. As we progress to epochs, the training loss decreases; however, a marginal rise in the validation loss is evident. This suggests an early indication of the model beginning to overfit the training data. Advancing to epochs 9-12, the training loss continues its descent, while the validation loss experiences a notable upturn. This serves as a robust indication of overfitting, where the model fits the training data too closely, potentially hindering its ability to generalize to new data.



(a) Loss over the epochs

(b) Confusion matrix

Figure 8: Classification results of BERT over the dataset with feature selection

To enhance the model's performance and promote better generalization, addressing overfitting is the key. Future improvements may involve incorporating batch normalization and max-pooling layers. Additionally, experimenting with different combinations of these techniques and fine-tuning hyperparameters (including dropout rates) could contribute to improved regularization.

4.2.7 Long Short Term Memory

In the exploration of LSTM (Long Short-Term Memory), the application of feature selection and integration with SenticNet yielded a 61% accuracy. Despite the efforts, achieving additional improvement in accuracy appears to be challenging under these conditions. Notably, the pre-feature selection phase presented its own set of challenges for LSTM, with the process becoming daunting due to the extensive feature set, surpassing 7000 in number. However, the outcomes, whether post or pre-feature selection, were ultimately unsatisfactory.

4.3 Classification 2: *Sarcasm* vs *Irony*

In this second classification task, the objective was to discern between irony and sarcasm in tweets that feature figurative language. It is crucial to highlight that this task did not follow the first one in sequence, as it did not operate on the same initial dataset. This decision was made to avoid splitting the original data twice, and to prevent the constraint of needing to achieve optimal results on the initial task, which could potentially compromise the validity of the second task. Consequently, we utilized the datasets from task 11 of SemEval-2015, which were already available and exclusively comprised figurative language. We applied the identical preprocessing pipeline described in Section 3.1 for the initial task.

Table 9 provides a summary of the accuracy outcomes on the test set obtained through Random Forest, Logistic Regression and Naive Bayes employing two distinct data encoding techniques. A comparison with the performance of BERT and RoBERTa is presented below.

	RF	**LR**	**NB**
Word Embedding	0.66	0.66	0.00
Count Vectorizer	0.72	0.73	0.74

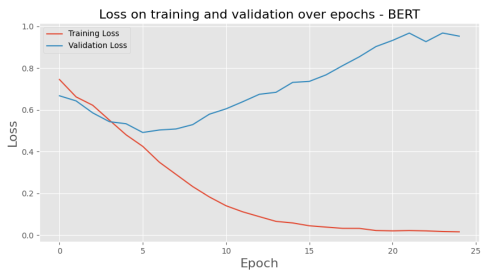
(a) Post Feature Selection

	RF	**LR**	**NB**
Word Embedding	0.68	0.72	0.00
Count Vectorizer	0.73	0.73	0.74

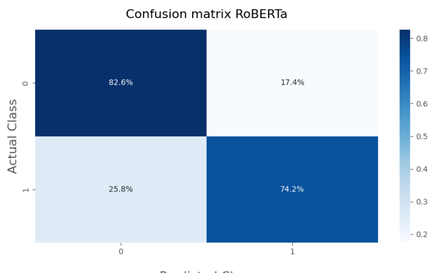
(b) Pre Feature Selection

Figure 9: Comparison of accuracy on the test set among different models trained on differently encoded datasets (RF: Random Forest, LR: Logistic Regression, NB: Naive Bayes). Table (a) shows the results post feature selecton, while (b) presents values pre feature selection.

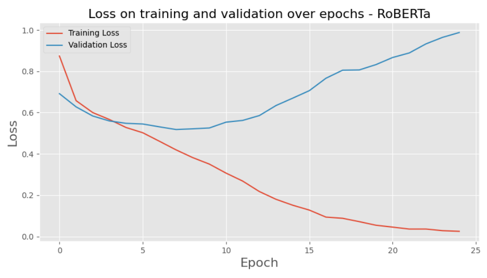
Similar to the approach taken in Classification Task 1, in this instance we opted to train BERT. Additionally, RoBERTa was utilized for this specific task [1]. RoBERTa is a model on which fine tuning has been done specifically for irony detection tasks. Figure 10 displays the progression of loss function values throughout epochs for both the training and validation sets, accompanied by the confusion matrices for each model. There is no noticeable significant difference in terms of loss. However, a closer examination of the confusion matrices indicates a slight decline in the model's capacity to identify class 1 when RoBERTa is employed.



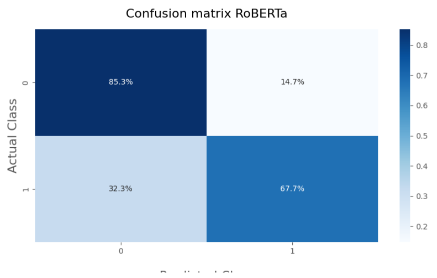
(a) Loss over the epochs - BERT



(b) Confusion matrix - BERT



(c) Loss over the epochs - RoBERTa



(d) Confusion matrix - RoBERTa

Figure 10: Classification results of BERT vs RoBERTa - second task

4.4 Explainability

Building upon the methodologies previously outlined, explainability task is performed. Figure 11 illustrates a decision tree: this serves as a visual representation of the intricate decision-making process employed by our model in classifying figurative language in tweets. Nodes within the tree depict specific criteria and thresholds used to partition the data, while leaves correspond to the final predicted class. In the figure, we can follow the classification path of a record present in the test set (local explainability). This visual depiction not only facilitates a clearer understanding of the decision paths but also highlights the critical features guiding the model's classification decisions.

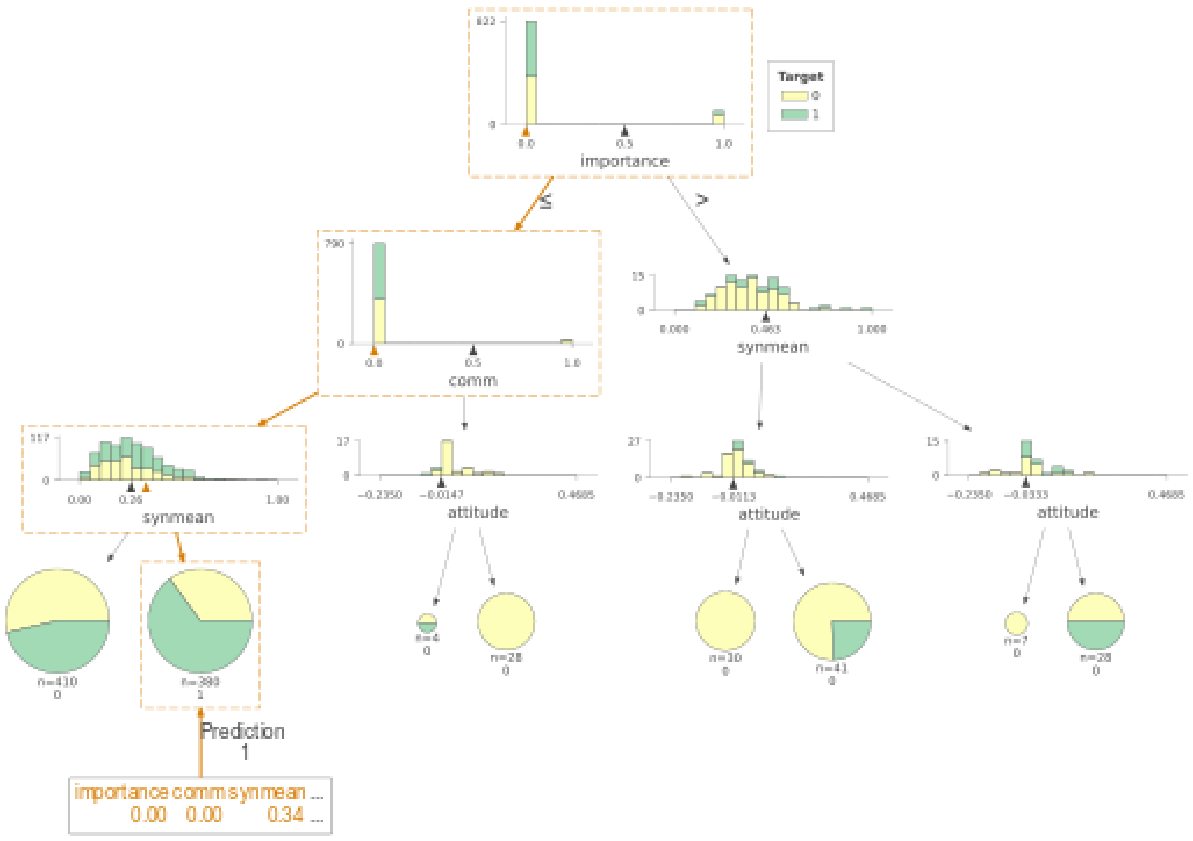
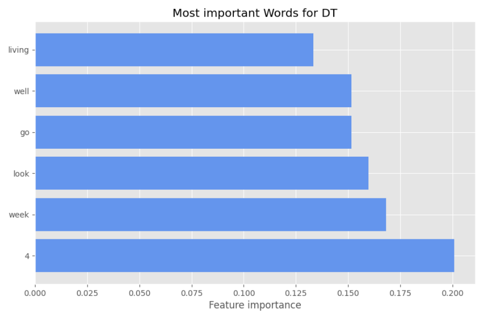
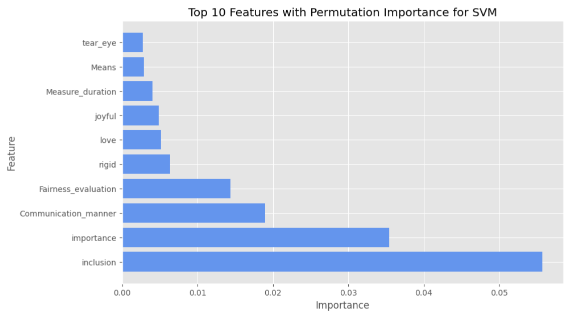


Figure 11: Partial Decision Tree trained on Feature-Selected Dataset

Further enriching our understanding of model interpretability, Figure 12 showcases two subfigures, each presenting horizontal bar plots representing feature importance. The first subfigure illustrates the feature importance analysis derived from the decision tree model, while the second subfigure provides insights into the feature importance within SVM model. The difference in the importance of features between the decision tree and SVM offers valuable insights into the diverse linguistic indicators that each model prioritizes when identifying figurative language in tweets.



(a) Feature Importance DT



(b) Feature Importance SVM

Figure 12: Most Important Features for Decision Tree and SVM

Further explorations on explainability have been conducted; specifically, we attempted to elucidate the predictions of one of the SVM models using an approximation of TREPAN. In practice, we trained a decision tree (intrinsic explainability) on the predictions of the SVM, treating the SVM as a black box to be explained (results are not reported here due to space constraints).

5 Conclusions

In summarizing our study on figurative language classification in tweets, we acknowledge that, despite employing a variety of models and analyses, our results may not be groundbreaking. However, beneath the surface, we identify valuable patterns, particularly the significance of trigrams (Figure 13) in shaping model performance.

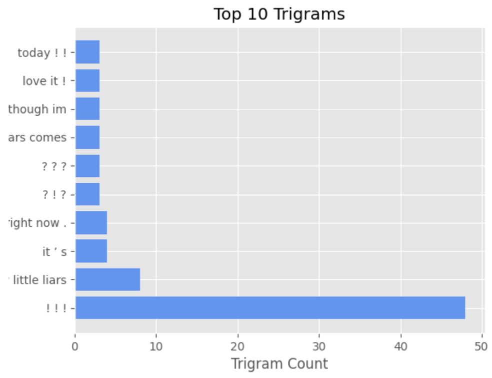


Figure 13: Trigrams

In this study we constructed a variety of datasets, extracting different structures and information hidden in the data and, on top of these ’versions’, we built and trained different models. We observed that our machine learning algorithms offered only limited improvement over the study conducted by Recupero et al.[3]. Starting from the conclusion of the paper, the accuracy obtained in their baseline experiments is congruent with our results. In the study and in our experiments, the introduction of semantic features corresponds to better accuracy in the classification, but this increment is not very marked.
Our empirical analyses highlight the potential for future improvements, particularly through the inclusion of trigram-based features. Additionally, our emphasis on thoughtful dataset enrichment and feature engineering, exemplified by the success of *Syno_Lower_Mean* and *Syn_Mean* features, underscores their crucial role in enhancing model performance. Empirical results from decision trees and SVM trained on this reduced feature space demonstrate commendable performance, accentuating the potential of such feature engineering strategies. While our current findings may not revolutionize the field, they lay the groundwork for future research. The potency of nuanced feature engineering and the identification of linguistic key patterns inspire further refinement of models and methodologies.

References

[1] Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online, November 2020. Association for Computational Linguistics.

[2] Francesco Barbieri and Horacio Saggion. Modelling irony in Twitter. In Shuly Wintner, Desmond Elliott, Konstantina Garoufi, Douwe Kiela, and Ivan Vulić, editors, *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 56–64, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

[3] Diego Reforgiato Recupero, Mehwish Alam, Davide Buscaldi, Aude Grezka, and Farideh Tavazoee. Frame-Based detection of figurative language in tweets [Application notes]. *IEEE Computational Intelligence Magazine*, 14(4):77–88, 11 2019.