UNIVERSITY OF COPENHAGEN
FACULTY OF HUMANITIES

Language Processing 2

Caroline Amalie Ørum-Hansen (ptq972), Maja Mittag (rzn202)
& Trine K. M. Siewartz Engelund (lcz713)

"Interesting 🤦‍♀️"

An Emoji-Based Approach to Profiling Irony and Stereotype Spreaders on Twitter

Date: June 13 2022

Abstract

Irony and stereotype spreading is not limited to spoken language – it exists on social media as well. Detecting people who spread this kind of information can be relevant for moderation or security purposes as well as for human-computer interaction. This paper presents an emoji-based approach to the IROSTEREO shared task on detecting irony and stereotype spreaders on Twitter at CLEF 2022. The goal of this author profiling task is to predict a binary label for each author. Our approach focuses on digital-specific phenomena such as emoji use as well as a special writing style based on a 2017-meme used to convey a mocking tone on text. We extracted 18 different surface-level features in three categories (stylometric, sentiment, and lexical), and built a pipeline with cross-validation and grid search that tested three classifiers (Support Vector Machine, Random Forest, and Logistic Regression). The resulting best model (Random Forest) obtained an accuracy of 0.892. An analysis of feature importance as evaluated by the Random Forest found that stylometric features were among the highest-ranked features.

Keywords — Irony detection, stereotypes, author profiling, emojis, sentiment analysis, PAN at CLEF 2022 (IROSTEREO)

The code can be found here: https://github.com/TrineSiewEngelund/LP2-exam-IROSTEREO

Division of Labour

All three authors have been equally involved in all decisions of this project and have contributed equally to both the experiment and the report. In the table below, we provide an overview of which section each author was main responsible for.

Section	Main responsible person
Abstract	Caroline, Maja & Trine
1.0	Caroline, Maja & Trine
2.1	Trine
2.2	Maja
2.3	Caroline
3.1	Maja
3.2	Caroline, Maja & Trine
3.3	Trine
3.4	Caroline
3.5	Caroline, Maja & Trine
4.0	Trine
5.0	Caroline, Maja & Trine
6.0	Caroline, Maja & Trine
7.1.1	Maja
7.1.2	Maja
7.1.3	Caroline, Maja & Trine
7.2	Caroline

Table 1: Main responsible person for each section

Table of Content

1 Introduction

Author profiling aims to classify authors into different classes based on their different writing styles (Oakes, 2021). A subtype of author profiling is the task of detecting irony and stereotype spreaders, which this paper deals with. Irony detection has gained interest over the past decade or so, especially due to the observation that e.g. sentiment analysis tends to perform worse on ironic texts than on non-ironic texts (Farias & Rosso, 2017). Other applications of irony detection include detecting irony and stereotype spreaders for security or moderation purposes, or human-computer interaction.

In this paper, we present our approach to detect irony and stereotype spreaders in Twitter feeds as part of the author profiling shared task (IROSTEREO) set by PAN for CLEF 2022 (PAN, 2022). The task at hand is thus to build, train, and test a model capable of determining whether a given author is an irony and stereotype spreader or not based on a selection of 200 of their tweets.

In addition to more traditional statistical and stylometric features, this study presents a couple of novel features. An example is our so-called SpongeBob feature, inspired by the Mocking SpongeBob meme from 2017 (Know Your Meme, 2022), that examines the proportion of words in mixed cases (i.e. a mix of upper- and lowercase used within the same word), e.g. "tHaT's BeCaUsE tHe MaJoRiTy Of AmErIcAnS aRe StUpId. KiNdA lIkE yOu".[1] Since the data comes from Twitter, where the style is informal and limited to 280 characters per tweet, various emoji features are examined as well.

The paper is structured as follows: First, the previous research, and a previous shared task on irony detection, are reviewed in section 2 to outline trends in the previous approaches and methods. Note that we review irony detection, which mainly deals with discriminating irony vs. non-irony on the document level, instead of the more general task of author profiling, since we are primarily interested in the relevant features used to detect irony. In section 3, the task and dataset from the current shared task are described, followed by a presentation of the preprocessing, feature extraction, and classifiers, we tested. Section 4 presents the results, which are then discussed in section 5. After the conclusion in section 6, answers to exam questions 2 and 3 regarding fake news and hate speech spreaders and emotion and affect lexicons can be found in section 7.

2 Previous research

2.1 Irony, Sarcasm, and Stereotypes

Multiple competing definitions of irony can be found in the literature. The most common definition, which we will adopt for our purposes, asserts that the meaning of an ironic utterance is

[1] This is an example from the shared task dataset.

the opposite of what is literally said (Barbieri & Saggion, 2014; Farias & Rosso, 2017, p. 113). Another common definition brings the context into play by defining irony as context incongruity, i.e. incongruity between a statement and its context (Zhang et al., 2019, p. 1634), while Grice's theory interprets irony as the speaker flouting the maxim of quality by saying what they believe to be false with the expectation that the listener will notice and understand this implicature (Garmendia, 2018, pp. 20-21; Farias and Rosso, 2017).

Related subjects include sarcasm, which some see as a subtype of irony (Farias & Rosso, 2017), and stereotypes, which includes a social bias about a specific group, e.g. women (Sánchez-Junquera et al., 2021). Hence, a stereotype is a type of generalization often based on a previous experience that leads to a belief or bias about a group of individuals (Kanahara, 2006). Irony and sarcasm, which are both instances of figurative language, are often used interchangeably in the literature. However, sarcasm differs from irony in that it is usually negative, aggressive, and offensive (Farias & Rosso, 2017). Even though this conceptual overlap between stereotypes and irony should be noted, we will only be dealing with irony and the stereotypes that might be spread alongside the ironic content in the rest of this paper as that's the topic of the IROSTEREO shared task (PAN, 2022).

2.2 Irony Detection

Irony detection is in its binary form the task of determining whether a given document is ironic or not – its main goal is to identify the features that can be used to discriminate between these two classes (Farias & Rosso, 2017). Multiclass irony detection exists as well but is beyond the scope of this paper. Irony detection is a relatively new task in NLP with one of the earliest papers addressing it being from 2009 (Carvalho et al., 2009; Farias & Rosso, 2017). A different approach is to detect the irony spreaders instead of the ironic documents, thus being a type of authorship profiling under the assumption that irony spreaders write differently than people, who don't spread irony. Authorship profiling also includes the related tasks of the detection of fake news spreaders and hate speech spreaders, which previous shared tasks from PAN 2020 and SemEval-2019 have dealt with (Basile et al., 2019; Rangel et al., 2020).

2.2.1 The Relevance of Irony Detection

One of the most obvious applications of irony detection is in sentiment analysis as it tends to perform worse on ironic texts compared to non-ironic texts since irony (as well as sarcasm) usually causes a polarity reversal (Farias & Rosso, 2017). An example could be "I just love being ill during the holidays", which appears to be positive, but is expressing a negative statement through irony. Furthermore, irony detection has practical relevance in areas such as human-computer interaction, chatbots, the detection of stereotype spreaders on social media, etc.

2.2.2 Typical Datasets and Types of Features

Irony detection can be carried out on different types of datasets, but they typically include social media texts from Twitter (Barbieri & Saggion, 2014; Reyes et al., 2013) or Reddit (Wallace, Charniak, et al., 2015), or customer reviews from Amazon (Buschmeier et al., 2014; Ravi & Ravi, 2017). Social media texts are characterized by being short, informal, and containing anomalies, emojis, and emoticons. In the previous research, the corpora used are either hashtag labeled, i.e. uses the author's tags such as "irony", or human-labeled, often through crowdsourcing (Farias & Rosso, 2017).

Farias and Rosso (2017) review previous research on irony detection. The most successful attempts use features, which can be grouped into the following categories: lexical, morphosyntactic, sentiment analysis features, semantic, structural, and stylistic. Thus, a lot of the successful approaches use surface-level features. Some work has been done on uncovering the useful features at the contextual (Wallace, Charniak, et al., 2015), affective and emotional (Farias et al., 2016) levels, but more efforts to do so could be taken according to Farias and Rosso (2017).

2.3 Previous Shared Tasks on Irony Detection

A past shared task is the SemEval-2018 "Irony detection in English Tweets" (Van Hee et al., 2018), which contained two subtasks – Task A was a binary irony detection task and task B was a multiclass irony detection task. Generally, the systems from task A performed better than those from task B.

THU_NGN (Wu et al., 2018) was the team that obtained the best results in task A with an accuracy of 0.735 and an F1-score of 0.705. They designed a densely connected Long Short-Term Memory (LSTM), which was based on word embeddings, sentiment features, and syntactic features. These syntactic features could for example be PoS-tag features and sentence embeddings (Van Hee et al., 2018).

The second-best team obtained an accuracy of 0.732 and an F1-score of 0.672 by constructing an ensemble classifier that consisted of two deep learning models: a word-based LSTM and a character-based LSTM. These models used pre-trained character and word embeddings (Van Hee et al., 2018).

The best performing team using unconstrained data[2] built an Support Vector Machine classifier that used various features and obtained an accuracy of 0.679 and an F1-score of 0.622. Structural features like hashtag counts, text length, and sentiment features were used to find the difference between the sentiment of the text and the emojis in a tweet. Additionally, they used emotion-based features defined by emotion lexicon scores (Van Hee et al., 2018).

In Evalita 2014, one of the subtasks was irony detection in Italian tweets (Farias & Rosso,

[2]That is, they used the shared task dataset as well as some additional data.

2017). Thus, each tweet was also labeled by whether it was ironic or not. Generally, the results from the irony detection subtask were lower than the other subtasks – the F1-scores were all under 0.60 in the irony detection subtask. The reason given for this was twofold. Firstly, irony detection was a more difficult task than sentiment analysis. Secondly, Italian NLP resources were lacking (Farias & Rosso, 2017). The most used type of supervised machine learning classifier was Support Vector Machines.

3 Methodology

3.1 The Task and Dataset

This paper presents our approach to the author profiling shared task at CLEF 2022 called "Profiling Irony and Stereotype Spreaders on Twitter (IROSTEREO)" (PAN, 2022). The goal of the task is to determine whether specific authors spread irony and stereotypes based on a sample of their Twitter feed. Hence, it is a binary classification problem.

The input data, which has been provided by PAN 2022 for the shared task, consists of 420 authors, who are each represented by 200 of their tweets in an XML file, and a ground truth label in a separate txt file that tells us whether she/he spreads irony (I) or not (NI). Thus, no ground truth label exists for each tweet, only for the authors. At least some of the irony spreaders also spread stereotypes about e.g. women or the LGBTQIA+ community but separating the irony *and* stereotype spreaders from those who only spread one or the other is not part of the task. There is a 50/50 distribution of irony spreaders and non-irony spreaders as each ground truth label occurs exactly 210 times. The expected output of the model for our purposes is a prediction for each author – do they spread irony or not? The mean length of all 200 tweets per author before preprocessing is 6,643 words (SD = 1,532) and 29,466 characters (SD = 7,157 characters).

3.2 Our Approach

In our approach, we chose to primarily focus on more novel features such as emoji use and tweet-internal sentiment differences. The reason for this is twofold: 1) Twitter employs a 280-character limit, thus making emojis a good way to economically express yourself, 2) irony is defined as expressing the opposite of what you mean. Our intuition is that this can be signaled in a tweet by expressing the opposite sentiment in the emojis than in the text, e.g. in the constructed example: "He's our best president so far 😧😩". In the remainder of this section, we will briefly explain how our approach was chained in a pipeline. Next, we will explain the preprocessing and the extracted features in greater detail.

3.3 Pipeline, Classifiers, and Evaluation

We used Scikit-learn's `pipeline` to chain the steps of our approach. Our pipeline was split into two branches of transformers (see Figure 1). In both branches, a preprocessor was applied first. The left branch concerned features in the shape of (-1,200) (i.e. 200 tweets for each author), while the right branch concerned features in the shape of (-1,1) (i.e. one document for each author containing all 200 joined tweets). The left branch included two different transformers extracting sentiment features, while the right branch included six different transformers extracting stylometric features. Since `pipeline` only accepts transformers which inherit from the Scikit-learn parent classes `BaseEstimator` and `TransformerMixin`, we built our transformers on top of these parent classes.

Using Scikit-learn's `FeatureUnion`, the output from the transformers was continuously combined, resulting in a sparse matrix. The sparse matrix was then scaled using `MaxAbsScaler`, which scales each feature by its maximum absolute value – and thus does not destroy the sparsity. Next, the data was randomly split in a train (80%) and test (20%) set. Based on previous research, we chose to train three different classifiers: a Support Vector Machine with a linear kernel (SVM), a Random Forest Classifier (RF), and a Logistic Regression (LR).

We will briefly summarize the algorithms behind the classifiers. SVM with a linear kernel aims to find the hyperplane that linearly separates the classes, so that the decision boundary is far away from the nearest training instance. SVM has two hyperparameters: `C` and `gamma`, which control the width and curves of the hyperplane (Géron, 2019, p. 153). Our second classifier RF is an ensemble of decision trees. Each decision tree is built up of nodes representing features. The number of trees is controlled by the hyperparameter `n_estimators` (Géron, 2019, p. 197-198). Our third classifier LR estimates probabilities for each class. It does this by computing a weighted sum of the input features plus a bias term, and then it outputs a probability of belonging to a class between 0 and 1 (Géron, 2019, p. 142-143). Like SVM, LR also has the hyperparameter C, which determines how well it should fit the training set.

To obtain the best hyperparameters for each classifier, the pipeline was fed into a grid search with 5-fold cross-validation using `GridsearchCV`. First, we did a coarse-grained grid search, and later on we narrowed the values to the one shown in Table 2. The models were then ranked based on their average accuracy in the cross-validation. Since accuracy is the evaluation metric for the shared task, we used that as well.

As a baseline, we trained an SVM on unigrams as proposed by the PAN shared task. This baseline received a test accuracy of 0.833. In the following sections, we will explain how the data was preprocessed and which features were extracted.

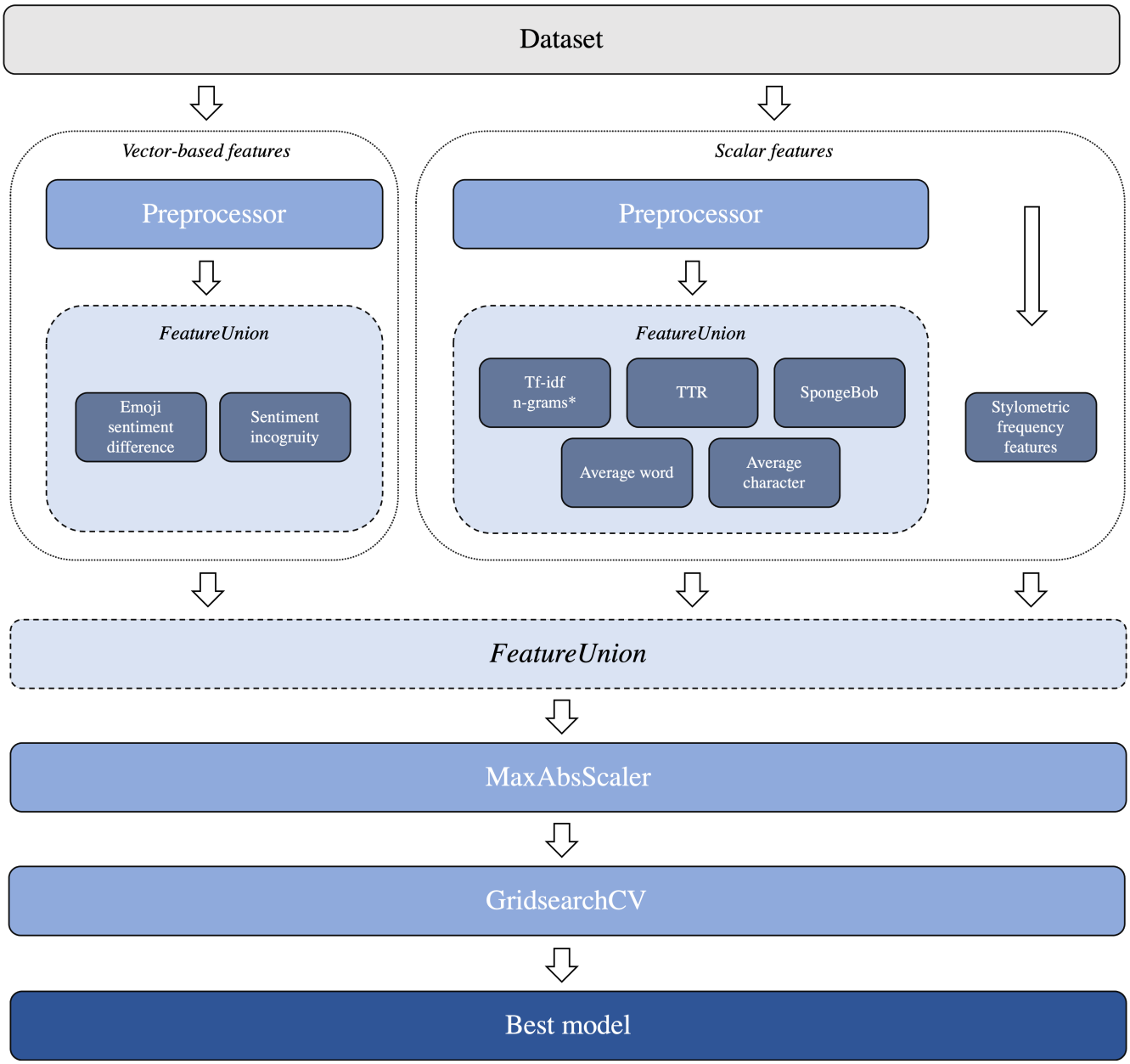

Figure 1: Pipeline of our approach

* Tf-idf is a vector-based feature, but uses the same preprocessor as the scalar features and are thus placed here.

3.4 Preprocessing

Anonymized tags (i.e., "#HASHTAG#", "#URL#", and "#USER#"), punctuation, and emojis were removed before extracting the scalar features to ensure proper performance of the tokenizer (`NLTK`'s `word_tokenize`), except for the stylometric frequency features, which counted the occurrences of these things amongst others. For that reason, the only preprocessing that was done for that group of features was lowercasing the text to catch all occurrences of the different laughs with the regular expressions (e.g. "haha" and "HAHA"). We didn't include lowercasing in the

Classifier	Hyperparameters	
	Name (Scikit-learn parameter name)	Values
Random Forest Classifier	Number of trees (`n_estimators`)	{200, 300}
Logistic Regression	Algorithm for optimization problem (`solver`)	{'liblinear'}
	Regularization coefficient (`C`)	{0.01, 1, 100}
Support Vector Machine	Kernel type (`kernel`)	{'linear'}
	Regularization coefficient (`C`)	{0.001, 1000}
	Kernel coefficient (`gamma`)	{0.1, 5}

Table 2: Parameter grid

preprocessing step for the other scalar features as the SpongeBob feature requires the original letter case since it measures the proportion of mixed letter cases in the tweets. Beside that, all the 200 tweets were joined into one document for each author. We did this to limit the run-time and the amount of high dimensional features.

As the presence of neutral terms like e.g. "HASHTAG" does not change the compound sentiment score found by `SentimentIntensityAnalyzer`, we decided not to do any preprocessing on the vector-based features, except for ensuring that all strings were non-empty by subbing empty strings with dots (`empty2dot`).

3.5 Features

We extracted 18 features from the tweets, which can be grouped into three overall categories: 1) stylometric features, 2) sentiment features, and 3) lexical features. An overview of the features can be found in Table 3. Most of the extracted features are quite common in the previous research, e.g. the various frequency features, and the tf-idf word n-grams ($n = 1$, 2, and 3). Noteworthy features include the SpongeBob feature, the six emoji frequency features as well as the two sentiment features. These will be reviewed in turn. It should also be noted that type-token ratios usually work better for documents of roughly equal length, but seeing as there are 200 tweets per author in the dataset, the number of tokens per author shouldn't differ too much to cause problems for the TTRs.

As previously mentioned, the SpongeBob feature is based on the Mocking SpongeBob meme from 2017, where a mix of upper- and lowercase letters are used within the same word to convey a mocking tone (Know Your Meme, 2022). One of the earliest examples includes the text: "Americans: I need healthcare because I have cancer and I'm dying. Republicans: I NeEd hEaLtHcArE bEcAuSe I hAvE CAncEr aNd iM dYinG" (Know Your Meme, 2022, Spread section). Even though this style of writing originated alongside a picture of the titular character from SpongeBob SquarePants, it is also used without the picture. Although mockery is not immediately linked

Feature(s)	Description
Stylometric features (scalar)	
Lexical diversity (TTR)	The Type-Token Ratio (V/N)
Mention frequency	The absolute frequency of "#USER#" mentions
Hashtag frequency	The absolute frequency of "#HASHTAG#"
URL frequency	The absolute frequency of "#URL#"
All emojis frequency	The absolute frequency of all emojis
Specific emoji frequencies ($n = 5$)	The absolute frequency of each of the following emojis: 🙂, 😂, 🤣, 🙃, and 👍 (i.e. five features)
Laugh frequency	The absolute frequency of different expressions of laughter, e.g. "lolol", "haha", "XD", etc. See Appendix A for the list that the regular expressions were based on
Repeated punctuation frequency	The absolute frequency of repeated punctuation, e.g. "?!", "!!!", "...", etc.
SpongeBob (mixed letter case)	The proportion of words that had a mix of upper- and lowercase letters (excluding cases, where only the first letter were capitalized)
Average number of words per tweet	The number of words in the tokenized string containing all 200 tweets per author divided by the number of tweets
Average number of characters per tweet	The number of characters in the tokenized string containing all 200 tweets per author divided by the number of tweets
Sentiment features (vector-based)	
Emoji sentiment difference	The absolute difference between the sentiment of only the emojis, and only the text from a given tweet
Sentiment incongruity	The absolute difference between the most positive and most negative expression in a tweet
Lexical features (scalar)	
Tf-idf n-grams	Word n-grams with $n = 1$, 2, and 3

Table 3: Extracted features

to irony, the shared task dataset contains stereotype spreaders as well, who might do so through mockery. The rationale behind our method for detecting words in this style is to count each word as 1 if it's neither in lowercase, uppercase, nor title case, thus excluding the forms "test", "TEST", and "Test", but including e.g. "teSt". The ratio between the number of mixed letter case words and all words is then returned.

An important part of our approach was emoji use and sentiment, which multiple features dealt with from different angles. Firstly, we extracted emoji use through six frequency features. One of

these counts all emojis used by an author through a regular expression pattern that includes all emojis up until 2020 (Gaitán, 2020, March 11), while the remaining five each count the occurrences of one of the following emojis: 🙂, 😂, 🤣, 🙃, and 👍. These specific emojis were chosen based on our intuition of ironic or mocking emojis as well as the conference project outlined in Pitenis (2021, December 12), which looked into whether each of 🙂, 😂, and 👍 was used the most in an ironic or non-ironic context. He found that while 😂 was ironic most of the time, it was also the emoji with the least agreement between the annotators – in general, the annotators only fully agreed 52.35% of the time (Pitenis, 2021, December 12).

Secondly, the emoji sentiment function returns the absolute difference between the sentiment of the emojis and the text in a given tweet using VADER – a value between 0 and 2. Lastly, the sentiment incongruity function returns the absolute difference between the most positive expression in a tweet and the most negative expression (a value between 0 and 2).

4 Results

For each classifier, we picked the model with the best hyperparameters as determined by the mean accuracy score in the 5-fold cross-validation process. The accuracies of these final models are shown in Figure 4. All three classifiers received similar mean accuracy scores during cross-validation. The RF Classifier received a mean cross-validation accuracy of 0.889, the LR an accuracy of 0.877, and the SVM an accuracy of 0.866. Thus, the RF Classifier is considered the best model, for which we obtained an training accuracy of 1 and a test accuracy of 0.892.

Model	Accuracy		
	Validation	Train	Test
Random Forest Classifier	0.889	1	0.892
Logistic Regression	0.877		
Support Vector Machine Classifier	0.866		

Table 4: Results

Figure 2 shows the confusion matrix normalized over the true labels of the test set compared to the predictions of the RF Classifier. It shows that there were no false negatives, i.e., all irony spreaders were correctly predicted to be irony and stereotype spreaders. On the other hand, there were 17% false positives, i.e., some authors were incorrectly predicted to be irony and stereotype spreaders.

To better understand how RF evaluated the importance of each feature, we computed the Gini importance facilitated by the `RandomForestClassifier` attribute, `feature_importances_`. This attribute outputs an array with a feature importance for each of the 82,115 features, which all

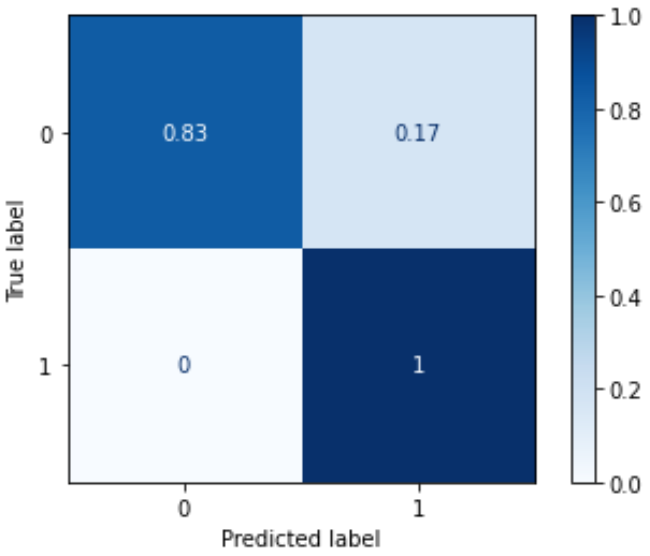

Figure 2: Confusion matrix normalized over the true labels (rows), where 1 = irony and stereotype spreaders, 0 = non-irony and stereotype spreaders

sum to 1. 81,699 were lexical features (i.e. tf-idf n-grams), 400 were sentiment features, and 11 were stylometric features. Since it would be too overwhelming to present the importance of all features, we selected a few that we found particular interesting to present in Table 5.

5 Discussion

This section first discusses the dataset provided for the shared task and its shortcomings, followed by the task at hand compared to earlier studies. Then, we compare our model to the baseline model and discuss the importance of some of the features. Lastly, we will alert future researchers about a specific Python package and make suggestions for future studies.

5.1 Dataset

A difficulty in this task lies in the dataset description or lack thereof since we don't know how the data was collected, labeled, etc. Is the corpus created through hashtag labeling, e.g. searching for tweets with the hashtag "#irony" and then selecting that author's last 200 tweets (or 200 randomly selected tweets?), or through a human annotation process – if so, to what extent did the annotators agree? Is it possible that one or more of the tweets by the authors labeled as non-spreaders is ironic or spread stereotypes? Furthermore, do the authors spread both irony *and* stereotypes or only one or the other? Seeing as sarcasm can be seen as a subtype of irony with a more negative goal (Farias & Rosso, 2017), one would imagine stereotype spreading to be more related to sarcasm than irony – and even more related to hate speech, which also aims to negatively impact certain groups based on their gender, sexuality, race, etc. (Anderson & Barnes,

Rank	Feature	Importance
1	Hashtag frequency	0.0072889
6	Mention frequency	0.0029689
15	Average no. of characters per tweet	0.0021767
34	Average no. of words per tweet	0.0015597
88	Emoji sentiment difference*	0.0009399
95	Lexical diversity (TTR)	0.0008775
101	Laugh frequency	0.0008319
179	URL frequency	0.0005931
187	Sentiment incongruity**	0.000576
1373	Repeated punctuation frequency	0.0001736
1645	Frequency of 👍	0.0001534
4590	Frequency of 😂	0.0000637
81974	SpongeBob	0
82111	Frequency of 🙂	0
82112	Frequency of 🤣	0
82113	Frequency of 🙃	0
82115	All emojis frequency	0

Table 5: Feature importance

* This is only the importance of the highest ranked emoji sentiment feature. All 200 features: $M = 0.0000980$, $SD = 0.000123$.
** This is only the importance of the highest ranked sentiment incongruity feature. All 200 features: $M = 0.0000984$, $SD = 0.000118$.

2022). All of these unanswered questions are very relevant to the interpretation of the relevancy and practical applicability of our system, yielding it difficult to provide any definite answers.

5.2 Author Profiling vs. Irony Detection

The relative difficulty of irony detection compared to other NLP tasks has been emphasized in previous shared tasks – recall that the highest accuracy and F1-score in the SemEval-2018 task were 0.735 and 0.705, respectively, while all the F1-scores in the Evalita 2014 were below 0.60 (Basile et al., 2014). However, our final model achieved a decent accuracy of 0.892. A possible explanation for this can be found in the slight differences in the goal of our approach and previous approaches, as they have primarily been concerned with determining whether a given document was ironic or not, whereas our goal was author profiling. Considering that we thus had more data per author than they typically had per document, this difference in performance makes sense. It should be noted, however, that while the model is relatively successful in correctly predicting the labels of the data, it's plausible that it's just really good at pattern recognition (and that spreaders

vs. non-spreaders exhibit different patterns in their tweets) rather than actually modeling irony and stereotype spreaders. In support of this pattern recognition hypothesis, it should be noted that even the simple baseline (SVM on unigrams) performed quite well, achieving a test accuracy of 0.833 and that the best performing features in the final Random Forest Classifier were surface-level features (see Section 5.4).

5.3 Our Model vs. the Baseline Model

Ideally, we would have wanted our best model to beat the baseline more convincingly. One explanation for our results could be that we implemented too many features on a relatively small dataset, resulting in a too complex model. When a model is too complex, it will have a harder time generalizing and thus make accurate predictions on unseen data (Müller & Guido, 2016, p. 28-30). This might be supported by the fact that our best model received a training accuracy of 1. On the other hand, our best model's accuracy for both the validation and test set was decent, which indicates that our model was able to generalize reasonably on unseen data.

One way to prevent overfitting is to add more data. Another strategy is to reduce the number of features by only selecting the most informative ones. This can be done with automatic feature selection. In one variant of automatic feature selection, the statistically significant relationship between the features is computed, before they are fed into a classifier. If a feature only adds information in combination with another feature, it will probably act as noise in the model, and therefore it can be discarded. In this way, only the most informative features are kept (Müller & Guido, 2016, p. 241). Due to time constraints, we did not execute this variant of feature selection.

Another variant of automatic feature selection is model-based. Here, the importance of each feature is measured as determined by a classifier. In this way, you can rank how essential each feature was for the model (Müller & Guido, 2016, p. 241-242). This is similar to what we did in section 4 when computing the feature importance as evaluated by the Random Forest Classifier. In the section below, we will discuss those results.

5.4 Feature Importance

In Table 5, we selected a few features of interest and listed them according to their rank as evaluated by the Random Forest Classifier. Due to the high number of features (82,115 in total), the feature importance of each feature was naturally very low.

Overall, our self-implemented stylometric transformers ranked high among the feature importances. 7 out of the 11 stylometric features were in the top 200. The emojis 👍 (rank 1,645) and 😂 (rank 4,590) contributed with some valuable information according to the Random Forest Classifier. Surprisingly, the all emojis frequency feature did not. A possible explanation of this could be that different types of emojis signal different things instead of e.g. a great amount of

emojis signaling ironic content. Instead, it might be better for future research to count the emojis in several categories such as "emojis depicting sentiment", "objects", etc.

Furthermore, the SpongeBob feature was evaluated to have zero feature importance. Since the Mocking SpongeBob writing style is a clear indication of mockery in internet culture (as described in section 4), we had hoped this feature would perform better. However, due to the recency of the origin of this specific writing style (2017) and the unknown age of the tweets in the dataset, the occurrences of this writing style might simply have been too rare in the dataset. Furthermore, the SpongeBob feature could have been designed as a vector-based feature instead of as a scalar feature, which might have made it more accurate than the current proportion seeing as the length of the tweets per author varies.

Concerning the two sentiment features, it was difficult to interpret the importances of those as each feature consisted of 200 features (one feature for each tweet). But since the best performing feature of each sentiment feature was ranked among the top 200, we cautiously interpret the sentiment features to have contributed some valuable information to the model.

Overall, the feature importances as evaluated by the Random Forest Classifier provided many insights into how our approach performed in the task of detecting irony and stereotype spreaders on Twitter. In the sections below, we will translate some of these insights into what could be done in future studies.

5.5 Recommendations for Future Studies

In our approach, emojis were one of the key features. Therefore, we started using the `demoji` package to find, remove, and decode emojis in the tweets. During the testing of our features, we discovered that this package as well as both the `emoji` and `emojis` packages took a long time to run. Later, we found out that VADER could actually measure the sentiment of emojis directly, instead of having to decode them using `demoji`, which we first did. Hence, we chose to discard `demoji` and use regular expressions to remove the emojis when necessary instead, since this change reduced the run-time to approximately 2% of the original run-time. Using this method, we are not able to capture all the emoji combinations from 2020 and onwards (Gaitán, 2020, March 11), but in this case the pros outweigh the cons.

In addition to recommending future studies to stay clear of the various emojis packages unless time is of no concern, we would urge future studies to look into more than surface-level features to get a better idea of what characterizes an ironic author, even though these work quite well as we have seen from our approach. An example could be contextual or emotional and affective features, drawing inspiration from e.g. the pragmatic contextual model in Karoui et al. (2015) or the affective lexicon approach taken in Farias et al. (2016).

Another important feature, which has been used in many previous approaches is word embeddings (see Basile et al. (2019) and Rangel et al. (2020) amongst others). An approach to include

the meaning of emojis is to train emoji embeddings (Barbieri et al., 2016). However, A. Singh et al. (2019) suggests a simpler method to process emojis compared to emoji embeddings. They suggest that you use the textual descriptions of the emoji that can be obtained by decoding the emoji for training regular word embeddings. Despite being very simple, this approach has proven effective by outperforming other approaches based on neural networks and emoji embeddings (A. Singh et al., 2019). There are two main reasons for this. When pretraining word embeddings, you have a massive number of words – in contrast to this, you don't have that many emojis. This means that the decoded emoji descriptions are more precise when they are "translated" to word embeddings compared to emoji embeddings. Another key detail is that the decoded description of the emojis can have words in common, which very likely wouldn't be translated to the emoji embeddings (A. Singh et al., 2019).

6 Conclusion

Our model performed satisfactorily compared to other studies with an accuracy of 0.892 on the test set. However, the current task was an author profiling task rather than an irony detection task, thus making direct comparison impossible. Speculatively, the author profiling task might just be easier, especially considering the high performance of the simple baseline. Furthermore, our model potentially contains too much noise because of superfluous features, which could have been prevented by e.g. automatic feature selection. The feature importance as evaluated by the Random Forest found that the best features were the frequency of hashtags and Twitter mentions. In general, we used surface-level features. Therefore, we would encourage future studies to look into pragmatic contextual features, as well as emotion and affective features.

7 Exam Questions

7.1 Question 2

7.1.1 Subtasks in Author Profiling

The IROSTEREO shared task (PAN, 2022), which this paper is concerned with, is an author profiling task, in which different authors' writing is analyzed to discriminate between two or more different classes (Oakes, 2021) — in this case, the task is to discriminate between irony and stereotype spreaders, and non-spreaders. Other examples of author profiling include the detection of fake news spreaders and hate speech spreaders, which past shared tasks have been concerned with (Basile et al., 2019; Rangel et al., 2020).

7.1.2 Usefulness of Our Stylometric Features

The aforementioned three subtasks of author profiling all share the same goal: to discriminate between binary classes of authors (spreaders vs. non-spreaders). Since the assumption is the same for the stylometric features regardless of the task – namely, that the authors in the two classes have different writing styles – we would expect some of our stylometric features to be helpful for fake news and hate speech detection as well. Of our stylometric features, we would expect the following to be useful in both of the other author profiling tasks: 1) lexical diversity as measured through the type-token ratio, 2) the frequency features (mentions, hashtags, URLs, emojis, repeated punctuation), and 3) the averaged tweet length in words and characters. We would also expect the tf-idf n-grams to be of help in both tasks.

Some of these stylometric features were indeed used by participants with well-performing systems in the PAN 2020 fake news spreaders shared task (Rangel et al., 2020).[3] The best performing model on the English tweets (Buda & Bolonyai, 2020) used various types of n-grams, various tweet-length statistics in both words and characters, the number of retweets, URLs, hashtags, mentions, emojis, and ellipsis at the end of the tweet, as well as the type-token ratio, which further supports our hypothesis that most of our stylometric features will be useful for detecting fake news spreaders as well as irony spreaders.

In the SemEval-2019 task, the first subtask was to detect hate speech against immigrants and women on Twitter both in English and in Spanish (Basile et al., 2019). In this task, various general stylometric features were used like tf-idf and different types of n-grams, which we also used in our approach to irony detection. Moreover, some participants in the SemEval-2019 task applied hate lexicons. In irony detection, this wouldn't be an obvious choice, because irony doesn't necessarily contain profanity or hate speech.

[3]Note that this task was multilingual (English and Spanish) as opposed to the IROSTEREO task.

7.1.3 Features Specific to Irony Detection

In contrast, there are certain features, which we don't necessarily expect to aid in discriminating between other types of information spreaders than irony and stereotype spreaders. Firstly, the two features with the sentiment differences (emoji vs. text sentiment, and most positive vs. most negative) are specific to irony in that they assume that these differences will be higher for irony spreaders than non-irony spreaders as irony contains a type of incongruity between the expression and the meaning, which is not necessarily the case for fake news and hate speech. Secondly, there's no obvious reason to suspect either hate speech or fake news spreaders of using less or more of the five specific emojis we looked into or expressions for laughter (such as "hahaha", "lmfao" etc.) than non-spreaders. However, there is a conceptual overlap between stereotypes and hate speech. Hence, we might suspect some of the irony-specific features to be useful for hate speech as well, e.g. the SpongeBob feature, which we would recommend doing for each tweet instead of at the author level.

7.2 Question 3

7.2.1 Emotion and Affect Lexicons

Emotion and affect lexicons are lists containing specific words and connotations, whose meanings are affective (Jurafsky & Martin, 2021). The simplest of these kinds of lexicons are binary in that there is a category for positive words and one for negative words, i.e. so-called sentiments (Jurafsky & Martin, 2021). If we look at some more advanced lexicons, they contain words with values for different affective dimensions. As an example, NRC Word-Emotion Association Lexicon (EmoLex) assigns eight basic emotions (Plutchik, 1980) and two sentiments (positive and negative) to the words with a binary value each (Jurafsky & Martin, 2021; S. M. Mohammad, n.d.).

Affect and emotion lexicons are built in a few different ways, e.g. by 1) human labeling, 2) semi-supervised learning, and 3) supervised learning. Nowadays you often use *crowdsourcing* for human labeling. Crowdsourcing means that you split your data and distribute the different pieces to many annotators (Jurafsky & Martin, 2021). Examples of human-labeled lexicons include S. Mohammad and Turney (2010) and S. M. Mohammad and Turney (2013). An example of semi-supervised learning is algorithm by Turney and Littman (2003). In this approach, you first collect positive and negative seed words, whereupon the distance from the seed words to every new word is measured (Jurafsky & Martin, 2021, p. 7). Lexicons built by supervised learning can be based on data from the internet that contains true labels, e.g. reviews with rankings. As an example, S. Mohammad (2012) used hashtags to capture emotion categories on Twitter to create a word-emotion association lexicon.

7.2.2 Usage in Natural Language Processing (NLP)

The usage of these lexicons in NLP varies. Examples include Abbasi and Chen (2007), who built an affect lexicon to perform affective analysis of extremist groups on the Dark Web, and a couple relevant studies, who utilized these kinds of lexicons for sentiment analysis (Hardeniya & Borikar, 2016; Suryadi, 2021). Lastly, Abd Al-Aziz et al. (2015) and N. Singh et al. (2019) used them for emotion detection.

7.2.3 Emotion and Affect Lexicons in Irony Detection

Affect and emotion lexicons are useful for expanding a sentiment analysis. Both Whitaker and Boonthum-Denecke (n.d.)'s and Maladry et al. (2022)'s approaches resemble our sentiment feature, where we compared the most positive and negative word in a tweet to each other. However, their approaches include emotion and affect lexicons, which results in more detailed sentiment analyses if the dataset is small (Jurafsky & Martin, 2021). Maladry et al. (2022) utilize the lexicons to capture "a clash between the sentiment of an annotated irony target and an explicit mention of the opposite sentiment, which is extracted from the remainder of the tweet based on the aforementioned sentiment lexicons" (p. 175). They report that using the lexicons to detect sentiment clashes was efficient. However, irony sometimes depends on the context, e.g. A says: "I just flunked P.E." B replies: "How nice". In these cases, emotion and affect lexicons as well as sentiment analysis might not be able to detect the ironic meaning.

References

Abbasi, A., & Chen, H. (2007). Affect intensity analysis of dark web forums. *2007 IEEE Intelligence and Security Informatics*, 282–288. https://doi.org/10.1109/ISI.2007.379486

Abd Al-Aziz, A. M., Gheith, M., & Eldin, A. S. (2015). Lexicon based and multi-criteria decision making (mcdm) approach for detecting emotions from arabic microblog text. *2015 First International Conference on Arabic Computational Linguistics (ACLing)*, 100–105. https://doi.org/10.1109/ACLing.2015.21

Anderson, L., & Barnes, M. (2022). Hate Speech. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy* (Spring 2022). Metaphysics Research Lab, Stanford University.

Barbieri, F., Ronzano, F., & Saggion, H. (2016). What does this emoji mean? a vector space skip-gram model for twitter emojis. *Calzolari N, Choukri K, Declerck T, et al, editors. Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016); 2016 May 23-28; Portorož, Slovenia. Paris: European Language Resources Association (ELRA)*, 3967–3972.

Barbieri, F., & Saggion, H. (2014). Automatic detection of irony and humour in twitter. *ICCC*, 155–162.

Basile, V., Bolioli, A., Patti, V., Rosso, P., & Nissim, M. (2014). Overview of the Evalita 2014 SENTIment POLarity classification task. *Proceedings of the First Italian Conference on Computational Linguistics CLiC-it 2014 and of the Fourth International Workshop EVALITA 2014 : 9-11 December 2014, Pisa*, 50–57.

Basile, V., Bosco, C., Fersini, E., Debora, N., Patti, V., Pardo, F. M. R., Rosso, P., Sanguinetti, M., et al. (2019). Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. *13th International Workshop on Semantic Evaluation*, 54–63.

Buda, J., & Bolonyai, F. (2020). An ensemble model using n-grams and statistical features to identify fake news spreaders on twitter. *CLEF (Working Notes)*.

Buschmeier, K., Cimiano, P., & Klinger, R. (2014). An impact analysis of features in a classification approach to irony detection in product reviews. *Proceedings of the 5th workshop on computational approaches to subjectivity, sentiment and social media analysis*, 42–49.

Carvalho, P., Sarmento, L., Silva, M. J., & De Oliveira, E. (2009). Clues for detecting irony in user-generated contents: Oh...!! it's" so easy";-). *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, 53–56.

Farias, D. I. H., Patti, V., & Rosso, P. (2016). Irony detection in twitter: The role of affective content. *ACM Transactions on Internet Technology (TOIT)*, *16*(3), 1–24.

Farias, D. I. H., & Rosso, P. (2017). Irony, sarcasm, and sentiment analysis. *Sentiment analysis in social networks* (pp. 113–128). Elsevier.

Gaitán, M. (2020, March 11). *EMOJI_PATTERN (comment)*. GitHub Gist. https://gist.github.com/Alex-Just/e86110836f3f93fe7932290526529cd1?permalink_comment_id=3208085#gistcomment-3208085

Garmendia, J. (2018). *Irony*. Cambridge University Press.

Géron, A. (2019). *Hands-on machine learning with scikit-learn, keras, and tensorflow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, Inc.

Hardeniya, T., & Borikar, D. A. (2016). Dictionary based approach to sentiment analysis-a review. *International Journal of Advanced Engineering, Management and Science*, *2*(5), 239438.

Jurafsky, D., & Martin, J. H. (2021). *Speech and language processing* (3rd ed. draft of December 29, 2021). https://web.stanford.edu/~jurafsky/slp3/

Kanahara, S. (2006). A review of the definitions of stereotype and a proposal for a progressional model. *Individual Differences Research*, *4*(5).

Karoui, J., Benamara, F., Moriceau, V., Aussenac-Gilles, N., & Belguith, L. H. (2015). Towards a contextual pragmatic model to detect irony in tweets. *53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*, 644–650.

Know Your Meme. (2022). *Mocking SpongeBob*. Know Your Meme. Retrieved June 6, 2022, from https://knowyourmeme.com/memes/mocking-spongebob

Maladry, A., Lefever, E., Van Hee, C., & Hoste, V. (2022). Irony detection for dutch: A venture into the implicit. *12th Workshop on Computational Approaches to Subjectivity, Sentiment & Social Media Analysis, collocated with ACL 2022*, 172–181.

McCulloch, G. (2015, May 15). *15 ways to laugh online*. Mental Floss. https://www.mentalfloss.com/article/63935/15-ways-laugh-online

Mohammad, S. (2012). #Emotional tweets. **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, 246–255. http://www.aclweb.org/anthology/S12-1033

Mohammad, S., & Turney, P. (2010). Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text*, 26–34.

Mohammad, S. M. (n.d.). *NRC word-emotion association lexicon*. Saifmohammad. Retrieved June 12, 2022, from https://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm

Mohammad, S. M., & Turney, P. D. (2013). Crowdsourcing a word-emotion association lexicon. *Computational Intelligence*, *29*(3), 436–465.

Müller, A. C., & Guido, S. (2016). *Introduction to machine learning with python: A guide for data scientists*. " O'Reilly Media, Inc.".

Oakes, M. P. (2021). Author profiling and related applications. *The Oxford handbook of computational linguistics* (2nd edition). https://www.oxfordhandbooks.com/view/10.1093/oxfordhb/9780199573691.001.0001/oxfordhb-9780199573691-e-53

PAN. (2022). *Profiling irony and stereotype spreaders on Twitter (IROSTEREO) 2022*. PAN at CLEF 2022. https://pan.webis.de/clef22/pan22-web/author-profiling.html

Pitenis, Z. (2021, December 12). *Annotating tweets as literal or ironic with LightTag*. zpitenis. https://zpitenis.com/blog/post/irony-detection-in-the-use-of-emojis/

Plutchik, R. (1980). A general psychoevolutionary theory of emotion. *Theories of emotion* (pp. 3–33). Elsevier.

Rangel, F., Giachanou, A., Ghanem, B. H. H., & Rosso, P. (2020). Overview of the 8th author profiling task at pan 2020: Profiling fake news spreaders on twitter. *CEUR Workshop Proceedings*, *2696*, 1–18.

Ravi, K., & Ravi, V. (2017). A novel automatic satire and irony detection using ensembled feature selection and data mining. *Knowledge-Based Systems*, *120*, 15–33.

Reyes, A., Rosso, P., & Veale, T. (2013). A multidimensional approach for detecting irony in twitter. *Language resources and evaluation*, *47*(1), 239–268.

Sánchez-Junquera, J., Chulvi, B., Rosso, P., & Ponzetto, S. P. (2021). How do you speak about immigrants? taxonomy and stereoimmigrants dataset for identifying stereotypes about immigrants. *Applied Sciences*, *11*(8), 3610.

Singh, A., Blanco, E., & Jin, W. (2019). Incorporating emoji descriptions improves tweet classification. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2096–2101.

Singh, N., Roy, N., & Gangopadhyay, A. (2019). Analyzing the emotions of crowd for improving the emergency response services. *Pervasive and mobile computing*, *58*, 101018.

Suryadi, D. (2021). Does it make you sad? a lexicon-based sentiment analysis on covid-19 news tweets. *IOP Conference Series: Materials Science and Engineering*, *1077*(1), 012042.

Turney, P. D., & Littman, M. L. (2003). Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems (TOIS)*, *21*(4), 315–346.

Van Hee, C., Lefever, E., & Hoste, V. (2018). SemEval-2018 task 3: Irony detection in English tweets. *Proceedings of The 12th International Workshop on Semantic Evaluation*, 39–50.

Wallace, B. C., Charniak, E. et al. (2015). Sparse, contextually informed models for irony detection: Exploiting user communities, entities and sentiment. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 1035–1044.

Whitaker, A., & Boonthum-Denecke, C. (n.d.). Integrating machine learning into irony detection model on twitter data.

Zhang, S., Zhang, X., Chan, J., & Rosso, P. (2019). Irony detection via sentiment-based transfer learning. *Information Processing & Management*, *56*(5), 1633–1644.

Appendices

Appendix A Laughter Expressions

The following list contains examples of different ways of expressing laughter in text. They are based on the examples provided in an article from Mental Floss (McCulloch, 2015, May 15).

Various versions of *lol*

- lol
- lolz
- lel
- lawl
- expanded versions, e.g. lolol, lollll, lolzz.

Various versions of *lmao*

- lmao
- lmfao
- expanded versions, e.g. lmaooo, lmfaooo.

Various versions of *rotfl*

- rotfl
- expanded versions, e.g. rotflol

Various versions of *haha*

- haha
- hehe
- heehee
- ahaha
- bahaha
- gahaha
- ahha
- mwahaha
- muahaha
- expanded versions, e.g. hahahaha, heheh.

Various versions of laughing or happy emoticons

- :)
- :')
- :D
- :'D
- XD
- xD
- =D
- expanded versions, e.g. :-D, :))