Language Resources and Evaluation
https://doi.org/10.1007/s10579-023-09656-1

ORIGINAL PAPER

The limitations of irony detection in Dutch social media

Aaron Maladry[1] · Els Lefever[1] · Cynthia Van Hee[1] · Véronique Hoste[1]

Accepted: 27 March 2023

Abstract

In this paper, we explore the feasibility of irony detection in Dutch social media. To this end, we investigate both transformer models with embedding representations, as well as traditional machine learning classifiers with extensive feature sets. Our feature-based methodology implements a variety of information sources including lexical, semantic, syntactic, sentiment features, as well as two new data-driven features to model common sense. Based on patterns in the syntactic structure of tweets, we aim to model the presence of contrasting sentiments, a phenomenon that is known to be indicative of verbal irony and sarcasm. Feature selection, as well as voting ensemble techniques were implemented to enhance the classification performance. The final systems reach F1-scores up to 0.79, which are promising results for a task as difficult as irony detection. Besides a quantitative analysis, this paper also describes a thorough qualitative analysis of the system output. Although lexical cues appear to be very important to express irony, our analysis also revealed the need for more advanced modeling of common-sense knowledge to detect more subtle examples of irony.

Keywords Irony detection · Sarcasm detection · Implicit sentiment modeling · Computational linguistics · Natural language processing · Machine learning · Neural networks · Language models · Social media

✉ Aaron Maladry
aaron.maladry@ugent.be

Els Lefever
els.lefever@ugent.be

Cynthia Van Hee
cynthia.vanhee@ugent.be

Véronique Hoste
veronique.hoste@ugent.be

[1] LT3, Ghent University, Groot-Brittanniëlaan 45, Ghent 9000, Belgium

Published online: 23 July 2023

1 Introduction

Although direct, clear and unambiguous language is highly praised in scientific literature, people are not always straightforward in their day-to-day communication and social interactions. People use figurative language for all kinds of creative purposes, be it to nuance or emphasize what they are saying, disguise their intentions or for sheer fun. By willfully violating Grice's conversational maxims (Grice, 1975) and providing literally wrong or useless information, people indicate that an utterance should be interpreted figuratively because there is a different implicit meaning. Figurative language is especially common on social media, where people are at liberty to express their thoughts as they like, for instance by using figurative speech. Although irony is a well-known and common example of figurative language, recognizing and understanding it remains a complex task. Hence it has been a popular research topic in the domains of linguistics, psycho-linguistics and, over the past decade, also in natural language processing (del Pilar Salas-Zárate et al., 2020).

When people say something ironically, they do not intend to convey the literal meaning of an utterance, but rather something else (usually the exact opposite). Observe the following example:

Example 1 *Aaah, don't you just love that awesome feeling* ***when you stub your stupid toe against the table*** :)

The lexical cues at the start of the utterance ("aaah, don't you just love") already give away that the tweet is intended ironically. However, even without these cues, one can still tell that nobody could be genuinely happy about stubbing their toe. This is a typical case of verbal irony, where irony as a figure of speech is realized in text[1]. In verbal irony, a person often expresses an exaggerated positive sentiment about an unpleasant or painful situation. This negative situation can be considered the "target" of the ironic evaluation, hence from now on, we will use the term "irony targets" to refer to these situations. Such a contrast between the sentiment of an evaluation on the one hand, and the implied or underlying sentiment of the target on the other hand, is known to be an important indicator of irony (Riloff et al., 2013).

In some cases, this sentiment contrast is clear in the text because the target is described with words that are inherently linked to a clear sentiment (such as "stupid" in this case). In many other cases, such explicit sentiment words are not necessarily present in the text and the reader needs common-sense knowledge to understand the implicit sentiment of the target in order to determine the sentiment contrast and consequently recognize the irony. As humans, we know which situations or events are pleasant or not because we likely experienced them ourselves. However, connecting this common-sense knowledge to a string of text is not trivial from a computational point of view, as language grows and continuously adapts to reflect our society or

[1] Based on previous research (Wilson & Sperber, 2012; Sulis et al., 2016) we consider sarcasm to be a specification of verbal irony, where sarcasm has a stronger negative connotation and is intended to ridicule, insult or hurt someone.

culture and because word connotations may be context-dependent. Combining strings into a longer sequence for instance, can alter the meaning of its constituents, e.g.: "walking your dog" generally has a positive connotation, while a longer target such as "walking your dog in the rain" becomes negative instead.

In the previous paragraphs, we explained how irony is verbalized from an intuitive human perspective. But to what extent can automatic systems detect irony in Dutch social media texts and which challenges still remain? Related research for irony detection almost exclusively focuses on English, but recently, the scope has extended to include more languages, such as French, German, Italian (Cignarella et al., 2020) and Arabic (Farha et al., 2022). However, it seems that the state of the art for low-resource languages such as Dutch still lags behind. Focusing on Dutch not only allows us to investigate to what extent methodologies for English can be ported to Dutch, but it also helps diversify the pool of researched languages and allows for more comparative future research.

To answer our main research question, we conducted an exhaustive set of experiments for Dutch irony detection using transformer-based architectures relying on text embeddings (Sect. 4) and SVM classifiers with a wide variety of features (Sect. 5). Both architectures are optimized and combined into an ensemble (Sect. 6). In addition, we present a novel approach to model implicit sentiment by detecting syntactic structures as irony targets and predicting their prototypical sentiment (Sect. 7). Besides a quantitative analysis, this paper also presents a thorough qualitative analysis of the output of the systems (Sect. 8). Through this manual evaluation, we gained insights on the performance, strengths and weaknesses of the different models. Finally, we summarize our findings, hypothesize about possible improvements to the state-of-the-art methodologies and present our suggestions for future research in the conclusion (Section 9).

2 Related research

As irony is an inherently subjective and complicated use of implicit language, its automatic detection is still far from perfect and therefore attracts much research attention. Early research into the detection of irony and sarcasm already recognizes the presence of lexical cues for irony detection. This was analyzed in detail by Kreuz and Caucci (2007), who suggest that lexical cues, usually in the form of interjections, rhetorical statements and formulaic expressions, could be a valuable tool for automatic detection. Although this pattern-based methodology had its merits and led to some of the first lexical (bag-of-word) approaches for automatic irony detection, it was only the first step in developing advanced feature-based classification systems. Davidov et al. (2010) and Tsur et al. (2010) worked on improving these lexical pattern features by considering the frequency and relevance of words (i.e. content words versus function words). Beside this, they expanded the feature set with syntactic features and trained a KNN (K Nearest Neighbours) model. Further additions included sentiment lexicon features and emoticons as affective sentiment information (González-Ibánez et al., 2011; Barbieri et al., 2014; Bouazizi & Ohtsuki, 2015). Although sentiment polarity information is confirmed to be particularly

helpful for irony detection, features referring to emotional dimensions (i.e. “fear”, “joy”, “anger”, with the exception of “love”) seem to play only a minor role (Farías et al., 2016).

Most of these features focus on identifying irony through the expression of an ironic evaluation. However, some researchers also investigated the target of these evaluations (i.e. what or who is the evaluation about?). Riloff et al. (2013) started modeling irony by searching for sentiment contrasts (i.e. positive against negative polarities) between an explicit evaluation, which could be either one out of 26 positive verb phrases or 20 predicative expressions they proposed as typical evaluations, and a described situation with implicit sentiment. Such situations were identified as a part of the text containing up to 3-grams that matched pre-specified syntactic patterns. This initial rule-based iteration of a sentiment clash was bootstrapped to an SVM with uni- and bi-grams. Van Hee et al. (2018) incorporated a broader version of the sentiment clash, which covers significantly more patterns and situations, into an extensive feature-based classifier. This system for English irony detection included not only lexical but also syntactic, sentiment lexicon and semantic (topic) features, which were proven useful in related research (e.g. [Liebrecht et al., 2013)].

While (Hee, 2017) was developing advanced feature-based models, Ghosh and Veale (2016) started introducing neural networks to the issue of irony detection. In their work, Long Short-Term Memory (LSTM) and convolutional neural networks (CNN) no longer used engineered features as input for their classifiers, but instead leveraged the information from word embeddings. For the irony detection task at SemEval 2018, researchers used both feature-based approaches with traditional classifiers, such as SVMs, Random Forest and Maximum Entropy, as well as neural architectures with word and sentence representations (Hee et al., 2018). While both methodologies were still in use, the latter (neural representation approaches) generally attain higher classification scores. The best performing competitors enhanced the input to their neural architecture by appending handcrafted features to the vector of static word embeddings. The features they employed most notably include sentiment and syntactic information (e.g. PoS-tags), as well as sentence embeddings (Wu et al., 2018).

With the dawn of Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018), using multi-headed self-attention (Vaswani et al., 2017) neural network architectures could better account for the surrounding context and were further optimized to make the most of embedding representations. Pre-trained BERT models that are afterwards fine-tuned traditionally start from tokenized text mapped to the model’s vocabulary and, after going through the hidden layers, end with a linear layer for text classification. To improve the performance of the final system, the embeddings of such pre-trained BERT models are often used as the input for other neural networks, similarly as this was done for previously mentioned static word embeddings. In recent research, transformer embeddings have been enhanced as input for neural networks (CNN, LSTM) in a variety of ways. While some techniques use a sequential approach and enrich the embeddings by first processing them through multiple neural architectures, others stay closer to the traditional approach and append information to the embedding vector. By making use of embedding vectors (Potamias et al., 2020) attempt to capture semantic

relations and encode contextual information with a CNN before classifying the enriched input with an LSTM. An example of the second technique is the study by Cignarella et al. (2020), who append syntactical information in the form of Part-of-Speech (PoS) n-grams to the embedding vectors with the intention to make a neural LSTM classifier more aware of syntactical patterns. Both approaches can also be combined. Babanejad et al. (2020), for instance, used two trained transformer models with a dedicated affective feature component to improve the contextual awareness of the basic embeddings while also adding affective sentiment information. Du et al. (2022) took this approach one step further by using one CNN to extract semantic features and combining those with contextual information as input for a second CNN to create emotional features. Next, all these features were used as the input to a bi-directional LSTM.

Combining different architectures and imbuing the system with additional (sentiment or syntactic) features has a positive impact on the classification scores. However, it is not always clear how the modifications improve the final system and literature describing the limitations of current systems is scarce. With the proposed research, we aim to investigate the feasibility of irony detection using both transformer models as well as feature-based approaches, and investigate the contributions of the various sources of information.

In previous work, we explored irony detection in Dutch social media from a quantitative perspective (Maladry et al., 2022). We provided the preliminary results for a traditional machine learning classifier combining syntactic, semantic, and lexical features with rule-based sentiment clash information. Additionally, these results were compared to those of the first BERT model fine-tuned on our data set. In this paper, we refine the methodologies and explore the benefits and limitations of varying feature groups and architectures by thoroughly analyzing the systems from both a quantitative and qualitative perspective.

3 Data description

For our experiments, we used the same corpus as in Maladry et al. (2022), which was originally gathered by Hee et al. (2016). The data was collected using the Twitter API, searching for tweets that contained irony-related hashtags, such as #sarcasme, #ironie and #not.[2]

The corpus consists of 5566 tweets and has a balanced label distribution. The non-ironic tweets (2783 instances) were posted by the same users who posted the ironic tweets. Although most labeled corpora are exclusively based on the presence of irony-related hashtags, this entire corpus was manually annotated by several annotators. This is relevant because 6% of tweets with an irony hashtag were annotated as non-ironic, and would thus receive the wrong label when labeled automatically. Out of 2783 non-ironic tweets, 217 (or 4% of the total data set)

[2] The methodology used to gather and annotate this Dutch corpus is exactly the same as for the corpus used in Task 3 at SemEval 2018 for irony and sarcasm detection in English (Hee et al., 2018).

still contained the irony hashtags, while the other 2566 do not contain any irony hashtags. As one might expect, the search strategy with irony hashtags has the downside that some of the annotated tweets were only clearly ironic because of the irony hashtag, like in this example:

Example 2 NL: *zo moeilijk is het toch niet? #sarcasme*

EN: *it's not that hard is it? #sarcasm*

As we aim to develop an irony detection system that does not rely on markers like hashtags, all irony hashtags are removed from all tweets for training and evaluation. Consequently, some of the ironic tweets could, technically, not be identified as ironic. For that reason, the annotators were instructed to indicate whether they considered the hashtag essential in order to identify a tweet as ironic (which was the case in 53% of the ironic tweets). There are a several possible approaches when working with such a data set. First, one could choose to ignore this downside and train and evaluate on all tweets. Secondly, one could remove all ironic tweets that did require the irony hashtags to be labeled correctly, which would lead to a corpus size reduction from 2783 ironic tweets to a mere 1308. Thirdly, one could train on all ironic tweets but evaluate solely on the ironic tweets that did not require the irony hashtag. Given that all of these approaches have their disadvantages, we opted for the first approach, keeping as much of the data as possible. Nonetheless, it is of paramount importance to take these irony hashtags into consideration when manually inspecting the system output. This way, we can optimize the amount of usable data and still account for the important downside of this search strategy (which does not have any efficient alternatives yet).

Besides differentiating between ironic and non-ironic tweets, the manual annotation allowed us to indicate different types of irony in the corpus. The first type of irony is probably the most well-known and most common form: verbal irony by clash. As was noted by Riloff et al. (2013), ironic and sarcastic tweets tend to contain a contrast between positive and negative sentiments. Usually, this is presented by an (unexpectedly) positive evaluation of a negative situation or event. For this type of verbal irony, the contrast is present in the text itself. The second type of irony is *situational irony*. This type of irony refers to something that happens in the real world that goes against the expectations. Often, these expectations are not literally included in the text and require common sense-knowledge. In some cases, a text can refer to the weather at the time or any kind of event (which may be hard to identify because only a reference is found in the text). The last type of irony was generally described as *other verbal irony* and intended to be a safety-net to catch types of irony that the annotator deemed ironic but could not fit in one of the two other categories. A tweet like "@someuser Gosh, this politician posted something stupid again, I didn't expect that. #sarcasm" includes an ironic contrast that is verbalized in the text but also requires some assumptions that require some situational knowledge or common sense.

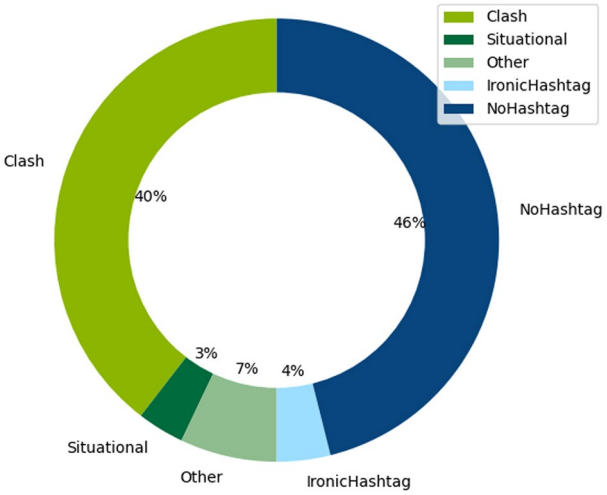

Fig. 1 Label distribution in percentage relative to the complete data set. Non-ironic labels are presented in blue, while ironic labels are shown in green

Most ironic tweets, 2201 or 79% of the ironic half of the corpus and 40% of the total data set, were labeled as ironic by clash. Situational irony is the least common type of irony in our data set and contributed only 190 tweets, 7% of the ironic tweets and 3% of the total data set. The remaining 392 tweets, 14% of the ironic tweets and 7% of the total data set, were tagged as other verbal irony. In Figure 1, we present an overview of the label distributions in our data set.

When indicating that a tweet was ironic by clash, the annotators were asked to select the "target" of the irony. In the first version of the data set by Hee et al. (2016) the targets were only annotated in a shallow form, as strings of text without any restriction in length or syntactic format. As a result, the annotators sometimes indicated the target as minimally as possible with only one or two words. At other times they opted for the longest possible interpretation of the target to make sure it was fully included. Since one of the goals of this paper is to take a first step in the direction of automatic extraction of these targets by using syntactic patterns, we decided to re-annotate all tweets that are ironic by clash to contain the longest and most specific target strings. In the ironic by clash set, 1524 targets were annotated, out of which 1511 were unique. For a large part (61%) of the tweets that were considered ironic by clash, annotators were able to indicate one or multiple irony target(s). This comes down to 24% of the entire data set containing such a target.

As test set across all experiments, we held out 20% (1113 tweets) of the total data. In the experiments with traditional classifiers, the remaining 90% (4453 tweets) was used as training data. We did not use a held-out development set for our traditional classifiers but used 10-fold cross-validation on the training set for optimization. When fine-tuning our transformer models, however, we did hold out 10% of the training data (445 tweets) as development set for optimization.

In the next section, we discuss our methodology for transformer approaches to showcase the performance of models with embedding representations without combining them with additional syntactic or affective information. Then, we explore how specific types of information (both grouped into thematic feature sets and combined) can be useful for irony detection in traditional statistical classifiers and as part of an ensemble approach.

4 Transformers architectures with embedding representations

4.1 Fine-tuned transformers

Since state-of-the-art results for text classification are often achieved by either transformer models or neural classifiers that can leverage the embeddings from those transformers, we fine-tuned a selected set of monolingual and multilingual models for irony detection in Dutch. In a first set of experiments, we used a Dutch monolingual BERT model, known as BERTje (Vries et al., 2019) and fine-tuned it for irony detection (Maladry et al., 2022). In our experiments, we expand upon this and evaluate both pre-trained monolingual and multilingual RoBERTa models. Below, we summarize the pre-trained models we tested for our experiments:

- BERTje (Vries et al., 2019), a monolingual Dutch BERT-base model trained with data from 5 corpora (one containing fiction novels (Books), two news corpora: TwNC (Ordelman et al., 2007) and Web news, the multi-genre SoNaR corpus without social media data (Oostdijk et al., 2013) and a Wikipedia dump from October 2019).
- Robbert (Delobelle et al., 2020), a monolingual Dutch RoBERTa model trained on the Dutch OSCAR corpus (a filtered CommonCrawl corpus) containing text from websites (not specifically social media).
- XLM RoBERTa—base (Conneau et al., 2019), the base version (with 250M parameters) of the multilingual RoBERTa model trained on a clean CommonCrawl corpus in 100 languages.
- XLM RoBERTa—large (Conneau et al., 2019), the large version (with 560M parameters) of the multilingual RoBERTa model trained on a clean CommonCrawl Corpus in 100 languages.
- XLM RoBERTa—Twitter (Barbieri et al., 2022), a model based on XLM Roberta-base that was further trained with 192M tweets. This is the only model that specifically includes Twitter data.

To counteract overfitting, we held out 10% of the training set and evaluated the model on this validation set after each epoch. As long as the F1-score on our validation set kept improving, we saved subsequent models. With this setup for fine-tuning, most of our pre-trained models reach close to their maximum validation F1-scores before epoch 15. In some cases, the model was not saved for a number of epochs before improving on the validation data. Table 1 presents the best results of each final model fine-tuned on the irony data set. The RobBERT

Table 1 Classification results for each of the fine-tuned transformer models

	Precision	Recall	F1	Accuracy	Epoch
BERTje	0.7101	0.7099	0.7089	0.7089	11
Robbert	0.7418	0.7419	0.7412	0.7412	5
XLM RoBERTa - base	0.7385	0.7377	0.7379	0.7385	15
XLM RoBERTa - Twitter	0.7537	0.7491	0.7493	0.7511	3
XLM RoBERTa - large	**0.7762**	**0.7706**	**0.7708**	**0.7727**	**12**

The scores are macro-averaged on the held-out test set of 1113 tweets

Bold signifies the highest scoring model (in 2 for each category)

and XLM RoBERTa - Twitter models reached their best F1-scores even long before the 10th epoch. We assume these models train faster because they have fewer "barriers" to transfer across. By this we mean a transfer from multilingual to monolingual, from general-domain to domain-specific or both. On the one hand, RobBERT was already pre-trained on Dutch monolingual data, so only needs to be adapted for the social media domain. On the other hand, XLM-RoBERTa - Twitter model was pre-trained on domain-specific data so only has to be adapted to Dutch. As Dutch is already one of the languages of the multi-lingual model, the transfer is not as hard, which is likely why it took even fewer epochs to fine-tune. In other words, transferring cross-domain had to be trained for 5 epochs and transferring from multilingual to monolingual only took 3.

Performance-wise, RobBERT outclasses BERTje by 3% F1-score. The multilingual XLM RoBERTa - base model does not outperform the monolingual RobBERT, even after fine-tuning for more epochs. This suggests that the inclusion of (possibly ironic) multilingual data during pre-training does not outweigh the advantages of training a language-specific model. Pre-training the model on the social media domain, for XLM RoBERTa - Twitter, does improve model performance for this task and even seems to have a more significant impact than keeping a model language-specific. We should mention, however, that the difference in impact may be overestimated as the domain-specific model is cross-lingual and does include Dutch. The language-specific models, on the contrary, do not include any data from the social media domain.

Finally, the best performing of these models is XLM RoBERTa - large. Fine-tuning of the additional parameters on our data set significantly amplified the model's ability to detect irony. A downside of this model is that it did take longer to train, since the model not only has more parameters to fine-tune but also takes around 10 epochs to reach the optimal weights. Furthermore, based on the results of the other pre-trained transformers, a language and domain-specific model with additional parameters (which currently does not exist) would likely be an even better fit for the task.

Table 2 Macro-averaged classification scores on the held-out test set for SVM and Logistic Regression with embeddings from pre-trained language models as input

	Precision	Recall	F1	Accuracy
Logistic regression				
BERTje	0.6383	0.6380	0.6381	0.6388
RobBERT	0.6457	0.6458	0.6457	0.6460
XLM RoBERTa - base	0.5801	0.5794	0.5773	0.5777
XLM RoBERTa - Twitter	**0.7044**	**0.7033**	**0.7034**	**0.7044**
XLM RoBERTa - large	0.6186	0.6186	0.6186	0.6190
SVM				
BERTje	0.6428	0.6427	0.6427	0.6433
RobBERT	0.6483	0.6483	0.6483	0.6487
XLM RoBERTa - base	0.5801	0.5794	0.5773	0.5777
XLM RoBERTa - Twitter	**0.6997**	**0.6990**	**0.6992**	**0.6999**
XLM RoBERTa - large	0.6087	0.6088	0.6083	0.6083

Bold signifies the highest scoring model (in 2 for each category)

4.2 Language model embeddings in statistical classifiers

An alternative approach to fine-tuning the embeddings from a pre-trained language model is to use these embeddings as input for a classifier. In related research, the classifier mounted on top of the embedding is a usually a neural network. Methodologically, we believe there's little difference between fine-tuning the weights of a neural classifier and those of a transformer, as long as no additional information, such as Part-of-Speech n-grams (Cignarella et al., 2020) is added to the embedding vectors. We instead propose a traditional classifier to leverage the sentence representations from the different pre-trained transformer models. Embeddings have already been used as input for an SVM before by Rohanian et al. (2018), as part of SemEval 2018 Wu et al. (2018). For an honest comparison to the fine-tuned models, we use the embeddings from the same pre-trained transformer models.

Our system setup includes two classifiers, logistic regression (LR) and a linear SVM. The logistic regression classifier was trained with a C-parameter of 0.1 and l2 (ridge regression) for regularization. The SVM is a linear SVM model with squared hinge loss, tolerance of e-05 and l2 penalty. These models were obtained by using TPOT (Le et al., 2020) as an optimization tool for both model and parameter choices. We ran the TpotClassifier genetic programming algorithm for 5 generations with a population size of 15 on our 10-fold cross-validated training set with weighted F1 as scoring metric. We optimized the models for the embeddings of each pre-trained transformer and found these two model for multiple source embeddings. Then, we trained and tested both optimized models for each of the embedding sources.

Table 2 tells us that the two classifiers, SVM and LR, perform comparably across the board, with LR, the simpler classifier, performing slightly better. As for the embeddings, those extracted with the domain-specific XLM RoBERTa - Twitter model were the most informative, followed by those of the language-specific models RobBERT and BERTje. The embeddings from XLM RoBERTa - large, the best

model for fine-tuning thanks to its high parameter count, performed poorly as the system does not seem to make use of the larger feature count.

In the end, integrating the embeddings into a traditional classifier results in a loss of about 7%, when comparing the best models from Table 2 to Table 1. For performance, domain and language-specificity of the pre-training data remain relevant factors for both system setups.

5 Traditional feature-based classifiers

5.1 Baseline system and feature engineering

Traditional machine learning approaches remain potent competitors to neural networks and transformer models, given that we can provide relevant features for the task. In Maladry et al. (2022) and Hee (2017), for example, a support vector classifier with an extensive feature set still achieved competitive F1-scores. For this paper, we experiment with further improving our baseline model for irony detection by expanding the feature set, while also filtering out the less relevant features with feature selection and optimizing different types of classifiers. As we also want to locate where our models still make mistakes, we do not limit ourselves to the search for the best-performing model, but complement this with more understandable classifiers. Some classifiers, such as the SVM algorithm with the rbf kernel, do not provide any feature importances or weights that indicate which features influenced the system's decision-making. Decision Tree-based systems, such as Random Forest, Bagging techniques and ExtraTree models tend to reach similar levels of performance but also provide insights into feature importance. To estimate both the highest quantitative performance and the influence of separate feature subsets, we set up experiments with two types of algorithms: SVM classifiers, which benefit more from high feature counts and a Decision Tree, which is a less complex approach that works better with fewer features but is more interpretable.

Lexical features constitute the largest feature subset, of which the n-gram features are the most basic ones. Our subset of 30,785 n-gram features combines both word n-grams for n-values of 1 to 4 and character n-grams for 3–4 (in both cases we set the minimal frequency to 3). Besides this, we count the occurrences of specific forms of creative language use, such as flooding in tokens (adding unnecessary vowels to a word i.e. "goood"), flooding in punctuation (duplicating punctuation, i.e. "..."), the use of hashtags, and fully capitalized tokens (i.e. "BAD").

The (shallow) *Syntactic* feature subset makes use of Part-of-Speech tagged input. For each type of tag, there is a binary feature to indicate whether the tag is present, a ternary feature (indicates whether it occurs "not", "once" or "more than once") and a frequency counter. In addition to the Part-of-Speech tags, we did the same for named entities (words referring to people, locations, organizations, etc.). Finally, we

included a more complex engineered feature named "temporal clash"[3]. This feature keeps track of the tense of each verb in a tweet and is triggered when a text is written in two different tenses, such as the past and present tense.

Sentiment Lexicon features check each tweet to search for polarity words based on existing sentiment lexicons. For each lexicon, the features keep track of the number of positive, negative and neutral words and also calculate a sentiment score for the entire tweet by summing the sentiment values of the individual retrieved words. This was done with the help of four lexicons that contain words, phrases, emoji's and emoticons: the Duoman Lexicon for subjective adjectives based on the NRC Jijkoun and Hofmann (2009), PATTERN (Smedt & Daelemans, 2012), the Hogenboom Emoticon Lexicon (Hogenboom et al., 2013) and the Emoji Sentiment Ranking (Kralj Novak et al., 2015).

The *Semantic* features capture the meaning of words instead of their form using information from background corpora. The first type of semantic features relies on creating Word2Vec (Mikolov et al., 2013) clusters that consist of words with similar meanings. One of the clusters that result from this contains "cursus" (EN: "course"),"academie" (EN: "academy"), "conferentie" (EN: "conference") and "lesgeven" (EN: "teaching"), which are clearly all words that are related to an educational or academic context. If a tweet contains a word that occurs in this cluster, the feature for this cluster is activated (i.e. its feature value is '1'). The 800 semantic clusters were generated with the Word2Vec (Mikolov et al., 2013) algorithm using a continuous bag-of-words approach, a vocabulary size of 100 words and a window size of 5, settings which were experimentally defined. While there are pre-trained versions of this algorithm, we trained our own model using a Twitter background corpus that better represents the social media domain. This background corpus consists of ironic tweets (i.e. tweets that contain one of the aforementioned irony hashtags) and non-ironic tweets, which lack such hashtags. The distribution of this corpus by label is 20% ironic and 80% non-ironic tweets.[4] In addition to the semantic cluster features, we included a second type of semantic features, to which we refer as the language model features. While the cluster features are used to capture the meaning of individual words, the language model features are used to represent the meaning of the tweets as a whole. To this purpose, we trained two separate language models using kenLM (Heafield, 2011). One of the models is trained exclusively on ironic tweets and the other only using non-ironic tweets. Both background corpora were gathered in the same way as was done for the cluster features. The two binary language model features then predict whether a tweet better fits the ironic language model or the non-ironic language model. Additionally, two binary features are included that check whether the tweet contains any out-of-vocabulary tokens for either of the language models.

[3] In past research, a similar feature has also been proposed by Reyes et al. (2013) and was used by Hee (2017).

[4] During pre-processing, the irony hashtags were removed from the tweets, as they are also removed from all train and test samples.

Finally, we propose two heavily-engineered variations of *Sentiment Clash* features. One variation is binary between positive and negative sentiments, while the other is a numerical feature that shows how different the sentiments are. In most cases, this clash occurs between an explicit expression of positive sentiment towards a target with an implicit negative sentiment. The primary challenge for targets carrying prototypical or implicit sentiment is that they are (obviously) not explicitly defined in a text and do not have clear boundaries. However, thanks to the manual annotations, we have an idea what irony targets look like semantically and syntactically. To determine the prototypical sentiment of a target string, we decided to consult Twitter as a potentially rich source of subjective content and estimated the general sentiment about the irony targets by gathering up to 500 tweets for each of our targets. This resulted in 1511 small background corpora. The prototypical sentiment was determined by performing sentiment analysis on the background corpora.[5] Finally, we counted the numbers of positive, neutral and negative tweets in each background corpus and used the distribution of these values as the representation of a target's implicit sentiment. In Maladry et al. (2022), the prototypical sentiment of a target was reduced to the most prevalent sentiment of tweets in the background corpus containing that target. In this paper, we converted the distribution of sentiment values for each target into a vector. The values [0.15, 0.20, 0.65], for example, imply that, out of all background tweets collected for this particular target, 15% are positive, 20% are neutral and 65% are negative. This indicates an overall (or prototypical) negative sentiment towards the concept. To compare these implicit sentiment vectors to an explicit [pos, neut, neg] vector, we search across the sentiment lexicons, counting the numbers of all positive, negative and neutral entries and turn these into a vector of the same format as the implicit sentiment distribution. Tokens that are a part of the annotated target string do not contribute to the explicit sentiment vector and no tokens are counted twice if they would occur in two different dictionaries. In case of tweets not having a single token present in any of the sentiment dictionaries, the explicit sentiment defaults to a null-vector with 0 as the values for positive, neutral and negative sentiments. To calculate the (now numerical) value of the sentiment clash, we measure the cosine distance between the explicit lexicon-based and the implicit data-based vectors. Since some tweets contain multiple targets, we consider the clash between each target and the explicit sentiment of the tweet without the target, and take the largest clash value as feature value for a tweet.[6]

[5] Some of these background corpora only contained a single tweet or none at all. To make the implicit sentiment detection more reliable, we arbitrarily chose to implement a minimum corpus size of five tweets. This reduced the final number of background corpora to 1014.

[6] We do not add up the different cosine distances because this would result in a large distance value for tweets with multiple targets. Even if we would average out the distances, this would still misrepresent the clash and reduce the significance of the biggest clash.

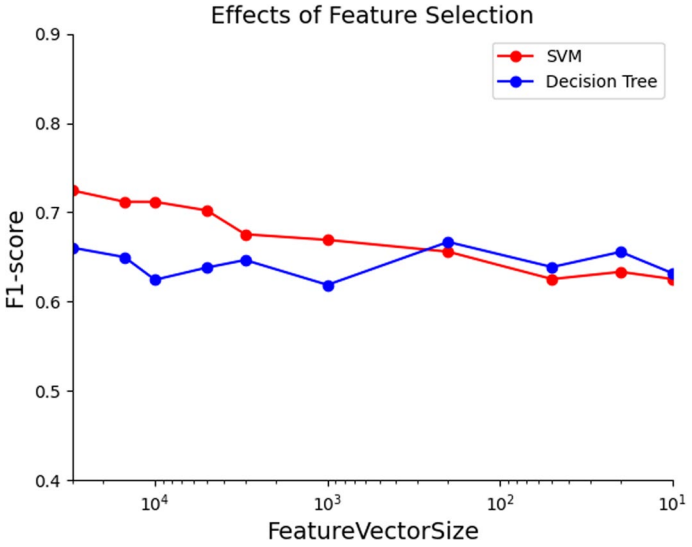

Fig. 2 Performance in F1-score for SVM and Decision Tree with gradually descending feature vector sizes (on log scale)

5.2 Feature selection

Given the large number of lexical features (>30,000), we determined feature informativeness using the mutual information metric. The primary goal of this selection procedure was to reduce the number of lexical features, but we extended the scope of our feature selection procedure to filter out all sub-optimal features. Since the clash features only have non-zero values for (some) ironic tweets, we already know that they have an informative value. As such, we would rather not optimize the system specifically for these two features and removed the clash features from the equation during feature selection and parameter optimization. Using mutual information for feature selection is likely not the optimal choice when optimizing the feature set for one specific model, but it remains a solid and computationally efficient method for model-agnostic feature selection.

The feature count was gradually lowered from 30,000 features down to the top 10 best features and tested for both an SVM classifier and a Decision Tree. For the SVM classifier, we scaled the feature values with a MinMaxScaler and used a C value of 2 with a gamma value of 0.0020 with the rbf kernel. For the Decision Tree, we l2-normalized the feature values and trained the model with a maximum depth of 4, we set the minimum sample leaves to 15 and minimum sample split to 15.[7]

[7] While we also experimented with grid-searching the optimal parameter values, the best cross-validation scores were attained by optimizing with TPOT (Le et al., 2020). The genetic tool optimized for 5 generations with a population size of 10 on a 10-fold cross-validation of our training set with weighted F1-score as the scoring metric.

Table 3 Macro-averaged classification scores on the held-out test set at different steps in our feature selection experiments

	Precision	Recall	F1	Accuracy
Systems without clash features				
30k SVM	**0.7249**	**0.7243**	**0.7244**	**0.7251**
30k Tree	0.6745	0.6668	0.6603	0.6631
200 SVM	0.6566	0.6566	0.6559	0.6559
200 Tree	**0.6728**	**0.6699**	**0.6668**	**0.6676**
Systems with clash features				
30k SVM	**0.7485**	**0.7487**	**0.7484**	**0.7484**
30k Tree	0.7752	0.7558	0.7477	0.7511
200 SVM	0.7296	0.7250	0.7215	0.7224
200 Tree	**0.7680**	**0.7487**	**0.7403**	**0.7439**

The F1-score of the SVM classifier did not improve by filtering out features with lower mutual information. This setup attained the highest F1-score (72.44%) using the complete set of 30,000 features. Decreasing the feature count gradually decreases the F1-score down to 62.52% when only the 10 best features are included, as shown in Figure 2. The Decision Tree does not follow the same behavior and achieves its highest F1-score (66.68%) at 200 features. Increasing the feature count for this model does not improve the system's F1-score. Compared to the SVM model, the Decision Tree performs better at all feature counts below 200 and worse when the feature count is over 200. The SVM, which is the more complex model, is able to leverage the additional information at higher feature counts, while the Decision Tree is confused by the addition of less relevant features.

The feature count of our SVM can easily be reduced to 5000, one sixth of the complete set, with only a minimal loss in performance of about 2%. At 5000 selected features, we are left with 4842 lexical, 18 syntactic, 20 sentiment lexicon and 120 semantic features. The lexical feature subset still makes up the majority of the features, followed by the semantic set. Taking this selection approach to the extreme and only using the 10 features with the highest mutual information, we still reached an F1-score over 60%, and thus a 10% drop in performance compared to the system with all features (Fig. 2).

The 10 features with the highest mutual information in our feature set are still mostly lexical (6 out of 10), however, they do represent some linguistic phenomena such as flooding ("ooo") and hyperboles ("such a", "awesome"). Besides that, another lexical features is an indicator of what used to be a hyperlink or image. The most relevant syntactic feature is the adjective count, which indicates that ironic tweets tend to use more adjectives, which we assume is essential for the contrasting positive evaluation of a negative situation. In a similar way, the positive tokens (and consequently the overall polarity score) from the PATTERN sentiment lexicon (Smedt & Daelemans, 2012) confirm that most ironic tweets seem positive on the

Table 4 Macro-averaged classification scores for Decision Tree and SVM systems for each feature group on the held-out test set

	Precision	Recall	F1	Accuracy
all lex SVM	*0.7077*	*0.7073*	0.7075	*0.7080*
all lex Tree	*0.6325*	*0.6320*	*0.6306*	*0.6307*
5k lex SVM	*0.6813*	*0.6795*	*0.6795*	*0.6810*
5k lex Tree	*0.6363*	*0.6352*	*0.6331*	*0.6334*
semantic SVM	0.6548	0.6549	0.6548	0.6550
semantic Tree	0.6872	0.6873	0.6872	0.6873
syntactic SVM	*0.5989*	*0.5898*	*0.5829*	*0.5948*
syntactic Tree	*0.5750*	*0.5739*	*0.5732*	*0.5759*
sentlex SVM	0.6161	0.5622	0.4979	0.5508
sentlex Tree	0.6216	0.6171	0.6111	0.6137
clash SVM	0.7880	0.6557	0.6029	0.6442
clash Tree	0.8026	0.6948	0.6574	0.6846

The models for each feature set are grouped together and are indicated in the same color

surface.[8] The most valuable semantic feature is the language model feature, which indicates whether the tweets are similar to the ironic tweets in our background corpus. We compared the best features based on mutual information to the those based on feature importance in the Decision Tree and found the same features were considered to be important.

After optimizing the parameters and performing feature selection, we finally also added the clash features (based on the full annotated target strings) as features for the Decision Tree and SVM models. The best Decision Tree (with 200 features + the two clashes) managed to catch up to the SVM (with 30k features + the two clashes), as shown in Table 3. In the next section, we evaluate the different feature sets separately and include the clash features (based on the annotated targets) as one of them.

5.3 Feature subset models

Mutual information only provides one perspective on feature importance. Different machine learning algorithms use different metrics besides mutual information to tweak feature weights and can thus perform better or worse with the same feature set. Since linguistic features are strongly intertwined / overlap, single feature importances might cause us to focus too much on individual words and overlook more general patterns. Based on mutual information, the lexical feature for "ooo", for example, is found to be an important feature. However, based on this feature, we cannot tell if it was important as an interjection, as flooding, flooding of an interjection or the actual lexical trigger for "ooo". To estimate how well the system can

[8] Maladry et al. (2022) hypothesized that a system for irony detection might look for positive tweets as a naive shortcut.

Table 5 Classification scores for ensemble models with and without the two sentiment clash features

	Precision	Recall	F1	Accuracy
Ensemble without clash features				
Full baseline	0.7105	0.7106	0.7105	0.7107
Only semantic and lexical	0.7185	0.7183	0.7184	0.7188
Semantic, lexical + XLM	0.7712	0.7699	0.7702	0.7709
All baseline + XLM	**0.7740**	**0.7726**	**0.7729**	**0.7736**
Ensemble with clash features				
Full baseline	0.7613	0.7606	0.7592	0.7592
Semantic and lexical /w clash	0.7800	0.7787	0.7771	0.7772
Semantic, lexical and XLM	0.7846	0.7835	0.7838	0.7844
Full baseline + XLM	**0.7842**	**0.7839**	**0.7840**	**0.7844**
SVM for reference				
SVM all features no clash	0.7249	0.7243	0.7244	0.7251
SVM all features /w clash	0.7485	0.7487	0.7484	0.7484

The baseline features are the lexical, semantic, syntactic and sentiment lexicon feature subsets, while XLM is the fine-tuned XLM RoBERTa - large model. The scores are macro-averaged on the held-out test set

Bold signifies the highest scoring model (in 2 for each category)

leverage the information in each of our feature groups, we trained classifiers for each of our feature subsets (lexical, syntactic, semantic, sentiment lexicon and our two clash features). Given the large size of the lexical feature set, it was reduced in size to a maximum of 5000 features.

In Table 4 the lexical features confirm their potency for irony detection, reaching an F-score of 71% with Support Vector Machines. Syntactic features and the sentiment word counts do not seem to be particularly useful, barely reaching F-scores of 60%. Surprisingly, the semantic features even bested the performance of the lexical features. While the SVM classifiers achieve better results with lexical and semantic features, the Decision Trees algorithms outperform them on the semantic, sentiment lexicon and clash feature sets.

6 Ensemble model

Since the SVM and Decision Tree classifiers benefit differently from the different feature groups, it could have merit to train individual models for each of the feature sets and combine the outputs of each model in an ensemble system. The simplest way to achieve this is to combine them with majority voting. This means that each model votes to indicate whether the tweet is ironic or not, making the label with the most votes the final result of such a system. However, there are quite some performance differences between the models trained on different feature groups, as shown in Table 4. Therefore, it makes more sense to weigh the vote of each system based on its classification performance on the training data. For the weights, we calculated

the average 10-fold cross-validated F1-score of each feature set system on our training set and replaced the binary classification output with the probabilities of the ironic / not ironic labels[9]. If a system with cross-validated F1-score of 69% would predict a tweet as ironic with a probability of 60%, the positive ('ironic') vote of that system would be 0.414 (0.69*0.60) while the negative vote ('not-ironic') would be 0.276 (0.69*0.40). Using the probabilities of the labels also takes the certainty of the system into account, resulting in a more nuanced voting system.

In the same way as we left out features with lower mutual information, we could also leave out feature groups that perform worse on average, which may delude the feature set as a whole. We tested this by evaluating ensemble systems where we excluded some of the feature groups from the equation. As shown in Table 5, the combined system for lexical and semantic features slightly outperforms the ensemble with all feature sets by less than 1%. However, if we compare this 71%, the best performance of the feature-based ensemble approaches, to the single SVM with all feature sets combined (Table 2), the latter still has a higher F1-score (72%).

Once we include the clash features as a separate feature group in the ensemble system, it not only catches up to the SVM with the complete feature set and clash (F1-score of 75%) but reaches 76%, as shown in Table 5. Most likely this is due to the fact that the informative sentiment clash features are no longer buried in the pile of less useful lexical features. Still, we should keep in mind that the sentiment clash features are currently still based on target annotations of ironic tweets. In Section 4.1, we also evaluated several fine-tuned transformer models and attained the best results with the pre-trained XLM-Roberta - large model. This best system can be combined with the feature-based approaches with the same voting strategy. Doing so further improves the scores of the ensemble model, pushing it to the first place in our system ranking, as shown in Table 5. This showcases that ensemble techniques are a valuable way to combine the strengths of multiple different models into a better final output.

7 Investigating syntactic parsing for automatic target extraction

In Sect. 5, we used the complete annotated targets to model a clash between the explicit sentiments of an ironic evaluation and the implicit sentiment of the evaluated target experience. In a realistic irony detection scenario, however, we should be able to automatically extract these targets. Once we are able to extract the targets to gather a background corpus for, we already have a working methodology to infer the implicit sentiment. Before attempting fully automatic target extraction, we first need to know what the targets look like syntactically. To estimate the syntactic characteristics of the targets, we propose a variety of patterns based on previous research.

[9] The Support Vector Machines algorithm does not work with direct probability, we used Platt-scaling to reach an approximation as it is implemented in sklearn.

Table 6 Coverage (in %) of the syntactic parsing approaches for the 1511 unique annotated target strings

Pattern	Coverage	Pattern Match Example
NounGroup	**0.7889**	[beautiful girl], [wild man]
Entities	0.1952	[Pope Francis], [John]
SVO	0.1754	[John kissed wife] [man hit keyboard]
VerbPattern	0.4408	[John kissed wife] [they danced on table]
VerbPhrase	0.4408	[bullying just continues] [and can't get by!]
Combined	**0.8068**	

Bold signifies the highest scoring model (in 2 for each category)

First, we syntactically parse all tweets into a dependency tree with spaCy (Honnibal & Montani, 2017).[10] Then, to find a match, we look for the intersection between the words in the tweet that match our syntactic pattern and the annotated targets strings. To illustrate this, the underlined words in the example below represent the words that match our syntactic pattern, while the words in bold were manually annotated as the target. If we take the **intersection** in Example 3, we only keep "stubbing my pinky toe" as the match for our pattern.

Example 3 *I love* **stubbing my pinky toe against the table**

If the words matching a pattern do not coincide with the annotated target strings, the clash feature values for this pattern are always 0. Analogously, if there are no annotated targets, the values for the clash features are zero as well. There are no annotated target strings for all non-ironic tweets and 45% of all ironic tweets. This means that, for this paper, we focus on the coverage that these patterns could provide, which only impacts the recall of the syntactic pattern approach. While the broadest patterns are likely to overgenerate potential targets, this will affect the precision of the approach, which is not yet taken into consideration.

Past research for English already explored some basic patterns which we used as inspirations and guidelines for our own patterns. Van Hee et al. (2018) used content words, dependency heads and Verb-Object as patterns. These patterns turned out to be noisy and too generic to provide reliable implicit sentiment for a sentiment clash. Joshi et al. (2016) used some very specific patterns for their rule-based extractor to detect the targets of sarcastic tweets. Some examples of patterns include named entities in addition to noun phrases containing positive adjectives, sentiment-bearing verbs and gerundial verb phrases. Our approach uses the insights of previous experiments to compromise between the very specific patterns of Joshi et al. (2016) and the broader patterns of Van Hee et al. (2018). We propose the following list of five syntactic patterns using universal dependencies (De Marneffe et al., 2021) and Part-of-Speech tags:

[10] We used the large news model for Dutch, available at https://spacy.io/models/nl .

- NounGroup: nouns with all noun and adjective descendants (children, grandchildren).
- Named Entities: people, organizations, locations, etc.
- SVO: Subject-Verb-Object in the exact order.
- VerbPattern: a broader form of Subject-Object-Verb. Starting from the verb, optionally adds the subject, adpositions (in, to, over, etc.), objects (both direct and indirect) and adverbs. This pattern skips most stop words (such as articles) and adjectives linked to the subject.
- VerbPhrase: all verb phrases consisting of each verb with its direct descendants in the parse tree. Whilst this overlaps a lot with the VerbPattern, this pattern is particularly useful for short tweets and robust against mistakes in dependency parsing.

As mentioned, the current goal is to maximize the coverage (i.e. recall) of our target extraction patterns. We present the coverage for each individual pattern and the combined set, along with some examples of pattern matches, in Table 6.

Out of 1511 unique annotated targets, the combined patterns were able to cover 1219 (81%). With these unique targets, we then inferred the implicit sentiment using sentiment analysis on the background corpora (cf. Section 5.1). Occasionally, the output of a syntactic pattern was nonsensical due to mistakes in parsing and POS-tagging as well as other unforeseen pattern matches. To filter these out, we doubled the lower bound of the corpus size for the annotated targets to 10 tweets. By doing so, we were able to provide an implicit sentiment for 1254 tweets. This is an improvement over the 1014 tweets with the annotated target patterns. The comparison between these setups is not entirely fair due to the increased lower bound, but if we had kept the same lower bound, the syntactic patterns would have covered even more targets.

With our pattern approach, we generate multiple matching targets (and implicit sentiments) for each annotated target. The target strings from these patterns often overlap. The SVO pattern, for example, always contains a NounGroup, because the subject is always a noun. More specific targets with more contextual information should better capture the essence of the annotated targets and have a more appropriate implicit sentiment. The word “dog” is probably considered positive because it refers to a cuddly pet. However, the more specific target “walking your dog in the rain” has a negative implicit sentiment. If the choice between these two is available, we would prefer using the implicit sentiment of the most specific target. Therefore, we also refined the results with a filter where we only use the most specific targets and to which we refer as *specific*. This filter removes each target that is fully included in another target for the same tweet.

The inferred implicit sentiments of these targets are then matched with the explicit sentiment representations of the remaining words in the original tweet to calculate the binary and numerical clash values. The experiment appeared to be very successful: the new targets generated both more numerical (+197) and binary (+111) clashes. To measure the impact of the sentiment clash features

Table 7 Classification scores (macro-averaged) for systems using only numerical and binary clash as features.

	Precision	Recall	F1	Accuracy
Decision tree				
Entities	0.7678	0.5591	0.4545	0.5628
NounGroup	**0.8050**	**0.7009**	**0.6655**	**0.6909**
SVO	0.7507	0.5348	0.3990	0.5193
VerbPattern	0.7679	0.5948	0.5083	0.5813
VerbPhrase	0.7663	0.5896	0.4995	0.5759
AllPatterns	**0.8154**	**0.7261**	**0.6981**	**0.7170**
AllPatterns - specific	0.8012	0.6913	0.6527	0.6810
Annotated	*0.8026*	*0.6948*	*0.6574*	*0.6846*
SVM				
Entities	0.2417	0.5000	0.3259	0.4834
NounGroup	**0.7685**	**0.5965**	**0.5112**	**0.5831**
SVO	0.7423	0.5026	0.3316	0.4861
VerbPattern	0.7567	0.5565	0.4408	0.5418
VerbPhrase	0.7559	0.5539	0.4359	0.5391
AllPatterns	0.7802	0.6330	0.5693	0.6208
AllPatterns - specific	**0.8009**	**0.6904**	**0.6515**	**0.6801**
Annotated	0.7880	0.6557	0.6029	0.6442

In this table, *All Patterns - specific* indicates that we filtered out the clash values for patterns that are included in other patterns

based on these syntactic patterns, we evaluated the performance with the same classifiers we used for the annotated clash subset in Section 5.3.

In Table 7, we present the results of the five individual patterns and both the refined and unrefined versions of the combined patterns as input for our SVM and Decision Tree classifiers. The improved coverage of the clash features results in a higher F1-score for the clash models. This improvement carries over to the ensemble model as well, where replacing the annotated clash model with the syntactically parsed clash model further improves the highest system performance from 78% to 79% F1-score. Removing the less specific targets actually hampers the performance of our system. To calculate the sentiment clash values, only the strongest contrast between implicit and explicit sentiments is kept for each tweet. This benefits systems with a larger number of implicit sentiments. This is also shown in the performance of the individual syntactic patterns. The pattern with the highest coverage, NounGroup, achieves the highest F1-score, while the patterns with the lowest coverage, named entities and SVO, perform the worst.

8 Manual error analysis

8.1 General overview

For our manual error analysis, we compared the annotated labels to the predictions made by the largest ensemble model, the three fine-tuned transformers (RobBERT, XLM RoBERTa - large and RoBERTa - Twitter), the separate feature subset models (as they were used in the ensemble) and finally the feature-based SVM classifiers (including and excluding the clash features both in annotated and parsed forms). The tweets were assessed in the same form as they are presented to the classifier models, anonymized and without any of the ironic hashtags. Additionally, we added the type of irony (ironic by clash, situational irony and other irony) and considered the annotated target strings for the analysis.

During the evaluation, we were once again reminded of the difficulty of this task. For some tweets, the irony even eluded the human annotators. In others, the text does not contain enough context information to determine whether a tweet is ironic. Example 4 was labeled as not ironic but in fact requires more context to determine if the label is correct:

Example 4 NL: *Bedankt voor de support op* ***mijn video van vandaag***! #Love

EN: *Thanks for the support on* ***my video today***! #Love

Similarly, tweets labeled as ironic can be open to interpretation as well. When the text itself does not suffice, people can make an educated guess to fill in the missing contextual information (temporal knowledge about previous events) or imagine a situation where one could say this ironically. As such, there is no clear upper boundary for irony detection but a fluid continuum where people or systems that share more common knowledge with the author of a text will be more suited to detect irony.[11] In Example 5, the original text was unambiguously ironic because the user added an irony hashtag (#not, #sarcasme or #irony) but the text has become ambiguous after removing the hashtag.

Example 5 NL: *Van je vrienden krijg je de beste adviezen.*

http://someurl.com
EN: *From your friends you get the best advice.*
http://someurl.com

In general, the results are satisfactory and some predictions are even impressive. To illustrate this, all of the classifiers correctly predicted Example 6 as being ironic.

[11] Common knowledge could be seen as being part of the same generation, cultural background, subcultures or even inside jokes.

The author compares the regional dialect of a local rapper on television to an enriching language immersion. Without such world knowledge, even a human would not be able to tell with certainty whether tweet is ironic.

Example 6 NL: *Na de opfrissing over de regel van drie*, ***nu ook nog*** *een snel taalbad AN met Slons****...*** *#reyerslaat*

EN: *after the refresher on basic arithmetic, we* ***now also get*** *a quick language immersion for standardized Dutch with Slons****...****#reyerslaat*

The ironic tweets are generally related to a number of topics, the most prominent of which are *politics*, *school*, *professional sports*, *public transport* and *the weather*. This topic information is exactly what we attempt to capture with the semantic cluster features. However, the list of relevant topics seems to be a lot shorter than expected, which suggests it might already suffice to create clusters based on a list of about five topics.

Taking a closer look at the data revealed that ironic tweets often contain **lexical triggers** that help to recognize the irony. When reading Example 6, one can imagine the author emphasizing and stressing these words when read out loud. Although these lexical trigger words or phrases do not contain any substantial meaning by themselves, they do contribute to the text by intensifying an evaluation and thereby giving away that it is intended ironically. Such lexical intensifiers and interjections hint at the possible ironic intention, but are by no means a definitive proof. As cultural function words, it is difficult to translate them.[12] Below, we present examples of words we consider to be lexical trigger words, their occurrences in the training corpus and how likely they are to occur in ironic tweets:

- *geweldig* (EN: great), 50 occurences (76% ironic)
- *fijn* (EN: fine/nice) 96 occurences (82% ironic)
- *lekker* (EN: fine/nice), 165 occurences (72% ironic)
- ...(suspension dots), 562 occurences (56% ironic)
- *toch* (EN: still/anyway), 205 occurences (64% ironic)

Some of these trigger words are even more likely to make a tweet ironic when they are used in conjunction with specific evaluation words.

- *wat een verrassing* (EN: what a surprise), 4 occurences (100% ironic).
- *goed idee* (EN: good idea), 4 occurrences (88% ironic).
- *weer gezellig* (EN: what a blast), 9 occurrences (78% ironic).
- *zoveel zin in* (EN: so looking forward to it), 6 occurrences (100% ironic).

[12] In Flemish Dutch, for example, we would probably not use “lekker” (literally “tasty” but also used as a positive evaluation of pretty much anything) to denote irony in the way people from the Netherlands do it.

Just like other strong, outspoken (and generally positive) evaluations, the exaggerations give away the true sentiment of the author. Therefore, without further context or common-sense knowledge, one could consider most expressions of intense sentiment or surprise as potential indicators of irony. These trigger words should already be captured by the lexical feature subset, which contains word and character n-grams occurring at least three times in the training corpus. While we provide an English translation of these trigger words, it remains to be investigated whether their occurrence in ironic utterances is similar to Dutch. However, this is worthy of a separate investigation.

8.2 False positives

Although exaggerated sentiments are important indicators for irony, they can just as well express an intense feeling. This makes genuine extreme sentiments prime candidates for misclassification. Based on Maladry et al. (2022), we expected the classifiers to look for abundantly positive sentiments as a shortcut for irony. This seems to be true to some extent, as we can see in the following non-ironic and wrongly classified tweets.

Example 7 NL: *GOOAAALLLL JAMES RODRIGUEZ 5-3!!!!*

Wat een geweldige wedstrijd!
EN: *GOOAAALLLL JAMES RODRIGUEZ 5-3!!!!*
What a great match!

Example 8 NL: *Lekker hard "No More Drama"*

van Mary J Blige meebleren! Heerlijk!
EN: *Bellowing along really loudly to "No More Drama"*
by Mary J Blige! I love it!

These examples suggest that the systems are using the intensity of the sentiment rather than its polarity. Although they are less common than the positive tweets, negative tweets, such as Example 9 and Example 10 were also predicted to be ironic while they were not.

Example 9 NL: *vind u echt zo een arrogant gastje he* >:-(

EN : *I really think you are such an arrogant dude* >:-(

Example 10 NL: *Een hele lange schooldag overleven.* >.<

EN: *Surviving a very long day at school.* >.<

The classifiers also make mistakes on non-ironic tweets with mixed sentiments. A potential reason for this is that the contrast between the positive and negative elements, based on the sentiment lexicon features, in the tweets is similar to the sentiment clash we are modeling in our experiments. In Example 11 and Example 12, the authors share their positive experience about a situation that would usually be considered negative (i.e. doing an exam and working in a nursing home). In Example 13, the contrast is of a temporal nature. After going through an annoying situation, the person shares his/her relief and positively evaluates the fact that the negative experience is over now or took a positive twist.

Example 11 NL: ***Waaah*** tentamen ging **best wel goed**

en nu heb ik ***vakantie*** *en dat is* ***leuk***
EN: ***Waaah*** exam went **pretty well**
and now I have ***vacation*** *and* **that's nice**

Example 12 NL: ***Complimenten*** van clienten maken je hele nacht **goed**

#nachtzuster #zorg #verpleeghuis **#liefde**
EN: ***Compliments*** *from clients make your* whole night **good**
#night nurse #care #nursinghome **#love**

Example 13 NL: *Vanmiddag wordt mijn* ***scherm gemaakt*** **#zielsgelukkig**

#al2dageninderouw
EN: *This afternoon my* ***screen will be fixed*** **#sohappy**
#beenmourningfor2days

8.3 False Negatives

Whilst the lexical triggers are clearly useful for irony detection, they only clarify the underlying irony of the utterance. Despite the presence of such triggers, the systems still miss the irony of some texts. As shown with Examples 14 and 15, these tweets contain verbal irony with an explicit ironic evaluation. However, determining the implicit sentiment and the exact target requires further reasoning based on complex common sense:

Example 14 NL: ***Heerlijk al die*** *#biologische #producten bij @alberthein in*

#plasticverpakking ***#dathelpt****hetmilieu*
EN: ***Delicious all those*** *#organic #products at @alberthein in*
#plasticpackaging ***#thathelps****theenvironment*

Example 15 NL: ***Dus goed idee*** *om alarm via internet/mobiel*

te doen in de toekomst #stroomstoring #geenservice
EN: ***Such a good idea*** *to do an alarm via the internet/mobile in the future #power failure #noservice*

To recognize the implicit sentiment, we need to know that plastic is harmful for the environment (Example 14), or that you cannot receive an online notification when you do not have service on your phone (Example 15). Alternatively, the hardest type of common sense requires knowledge about conversations and insight into what people would realistically say or not. For Example 16, we know every citizen has the right to vote, but we would never specifically emphasize that "this person probably" has that right, as this is a well-known fact relying on world knowledge.

Example 16 NL: *@someuser en dit figuur heeft* ***waarschijnlijk ook gewoon*** *stemrecht...*

EN: *@someuser and this figure* ***probably also just*** *has the right to vote...*

8.4 Transformers versus feature engineering

The manual analysis did not reveal clear differences between the type of tweets misclassified by transformers or feature-based approaches. Both systems seem to rely on intensified sentiment (often presented by expressive and emphasizing lexical triggers) to classify a text as ironic. Still, they struggles to identify whether the sentiment in such an intense utterance is genuine or ironic.

We did, however, notice some minor differences between both approaches. It seems that the feature-based approach depends more on detecting trigger words. This hypothesis is supported by the fact that tweets that contain flooded trigger words are not detected as ironic. Despite the fact that there is a feature that detects flooding (an alternative intensifier), this does not compensate for the missing lexical feature. Because of this, the following tweets were wrongly classified as being non-ironic:

Example 17 NL: ***Gooodmorning***, *vandaag introdag*, ***wat leeeuk***

EN: ***Gooodmorning***, *today introduction day,* ***how niiiiice***

Example 18 NL: *@someuser ik maak huiswerk* ***yessss***

EN: *@someuser I'm making homework* ***yessss***

At the same time, the feature-based approach does miss out on some tweets that contain obvious lexical triggers. Example 19 contains the lexical trigger "lekker" (EN: nice), which appears in ironic tweets for 66% of its 186 occurrences in the train

set. Similarly, the lexical triggers "wel weer" (EN: sure) are key to conveying the irony in Example 20. As we mentioned before, such n-grams should be a part of the lexical feature set as long as they occur at least 3 times in the training set.

Example 19 NL: @*someuser* ***Lekker*** *dan.*

EN: @*someuser* ***Well nice***.

Example 20 NL: *Nu.nl weet* ***wel weer*** *prioriteiten te stellen.*

EN: *Nu.nl* ***sure*** *knows how to prioritize.*

8.5 Value of sentiment clash

The sentiment clash features are one of the focus points of this paper. To investigate to which extent these features improve model performance, we compare the results of the systems with clash features to the systems without clash features. The prototypical use-cases for the clash feature contain an **explicit evaluation** and have a target with a clear implicit sentiment. To our surprise, most of these ideal use cases were already classified correctly by all systems without the clash features, as shown in Example 21 and Example 22.

Example 21 NL: *Het ene grote bedrijf na de andere gaat #failliet.*

Lang leven investeringsmaatschappijen.
EN: *One big company after another goes #bankrupt.*
Long live *investment companies*.

Example 22 NL: *#D66 is voor Amerikaans systeem. Iedereen straks 2-4 baantjes.*

En nog niet kunnen rondkomen! ***Joepie!*** *#wnl*
EN: *#D66 is for American system. Everyone soon 2-4 jobs.*
And still can't get by! ***YAY!*** *#wnl*

However, lexical triggers do not always suffice to detect irony. Despite the two clear trigger words "Fijn!" and "Handig", the systems without the clash features (including the fine-tuned transformers) did not catch the irony of Example 23.

Example 23 NL: ***Fijn!*** *iTunes afsluiten maar de muziek speelt wel verder*!

#Handig #SluitAfTrut
EN: ***Nice!*** *Close iTunes but the music just keeps playing*!
#Useful #CloseBitch

Due to the explicit positive evaluation and implicit negative sentiment, the systems with the clash features were able to identify a sentiment clash and detect the irony. Whilst the lexical triggers make the irony of a situation more explicit and serve a supportive role in the expression of irony, the actual irony does rely on world knowledge of the target situation. In spoken language, the irony is made explicit through tone of voice and emphasis. Since these tools are not available in the text medium, we assume the lexical triggers serve as an alternative way to clarify the irony.

Sometimes, understanding the target requires complex numerical common sense. When specific numbers occur in the target, the data-driven approach does not generalize enough to determine the implicit sentiment, as illustrated by Example 24.

Example 24 NL: ***Wauw*** *ik ken* **al** *15 van de 110 woordjes en nog geen zinnen*.

Gaat geweldig.
EN: ***Wow I already*** *know 15 of the 110 words and no sentences yet*.
Going great.

9 Conclusion & future research

In the presented experiments, we thoroughly examined a variety of systems, ranging from state-of-the-art transformer models to more traditional machine learning with feature-based classifiers. Our experiments show that fine-tuning a language and preferably also a domain-specific transformer model achieves the best single-model results, with improved results at higher parameter counts. Our best performing single model was a fine-tuned XLM-Roberta-large model, which attained an F1-score of 77%. Nevertheless, a more traditional feature-based SVM still reaches competitive results (72% without clash features). After assessing the different feature groups, we concluded that lexical and semantic feature groups are the most valuable sources of information.

Additionally, we developed two sentiment clash features that encode pertinent common-sense knowledge for irony detection. We infused our traditional approach with our sentiment clash features (reaching an F1 score of 75%). We then further combined all models, including not only feature subsets but also the fine-tuned transformer model, into a weighted ensemble model to achieve an F1-score of 78%. Using syntactic patterns, we optimized the coverage (i.e. recall) of our sentiment clash features and, once included into the ensemble, attained our best system performance (79%).

The manual error analysis solidifies the importance of the lexical feature set. All systems seem to rely heavily on recognizing iconic lexical triggers. However, the same triggers words are often also used to intensify sentiment. Consequently, genuine strong positive or negative sentiments were occasionally mistaken for irony. During our manual error analysis, we also investigated the usefulness of our sentiment clash features. This revealed that the prototypical tweets containing

clear verbal irony, for which we engineered sentiment clash features, are often already identified without those features. The sentiment clash features, did, however, improve the system when the situations or evaluations were more subtle. The lexical trigger mostly helps to make the ironic intention of the evaluation more explicit, which is especially useful for text, where vocal cues and emphasis are not available. In some tweets, the evaluated situation is hard to identify in the text. These cases (annotated as "other verbal irony") often require complex common-sense reasoning.

For this work, we only considered the recall of our syntactic pattern approach, which is still anchored in the annotated targets. For future research, the syntactic pattern approach will be further developed in order to enable automatic extraction of evaluation targets from ironic and non-ironic tweets, while taking into account a good balance between recall and precision. Using the resulting targets as potential common-sense units, we aim to build new knowledge bases for any given domain, that can be further expanded and enhanced by adding additional reasoning mechanisms to create connections between the common-sense units. While we relied on our manual evaluation of the system outputs to investigate the limitations of our systems, it would also be interesting to investigate how techniques based on attention weights (Abnar & Zuidema, 2020),game theory (Fernando et al., 2019) or integrated gradients (Sundararajan et al., 2017) can help explain the reasoning of transformer models. Such techniques can help us identify the "features" used by transformer models and possibly establish analogies between the reasoning of traditional feature-based and transformer-based systems.

Author contributions AM: wrote the main manuscript, performed the experiments and created the figures for the manuscript. VH: is the primary supervisor for the PhD project of AM. She reviewed the manuscript and advised on the experiments. EL and CVH: are co-supervisors for the PhD project of AM. They reviewed the manuscript and advised on the experiments.

Funding This work was supported by Ghent University under grant BOF.24Y.2021.0019.01 as well as the interdisciplinary NewsDNA project under grant BOF.GOA.2018.0006.03.

Data availability The data will not be made publicly available unless requested. The experimental data was anonimized and remains subject to the Twitter Privacy Policy.

Code availability Not available.

Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

Ethical approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

Abnar, S., & Zuidema, W. (2020). Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (pp. 4190–4197). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.385. https://aclanthology.org/2020.acl-main.385

Babanejad, N., Davoudi, H., An, A., & Papagelis, M. (2020). Affective and contextual embedding for sarcasm detection. In *Proceedings of the 28th International Conference on Computational Linguistics*, (pp. 225–243). International Committee on Computational Linguistics, Barcelona. https://doi.org/10.18653/v1/2020.coling-main.20. https://aclanthology.org/2020.coling-main.20

Barbieri, F., Espinosa Anke, L., & Camacho-Collados, J. (2022). XLM-T: Multilingual language models in Twitter for sentiment analysis and beyond. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, (pp. 258–266). European Language Resources Association, Marseille. Retrieved from https://aclanthology.org/2022.lrec-1.27

Barbieri, F., Saggion, H., & Ronzano, F. (2014). Modelling sarcasm in twitter, a novel approach. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, (pp. 50–58)

Bouazizi, M., & Ohtsuki, T. (2015). Sarcasm detection in twitter:" all your products are incredibly amazing!!!"-are they really? In *2015 IEEE Global Communications Conference (GLOBECOM)*, (pp. 1–6). IEEE

Cignarella, A.T., Basile, V., Sanguinetti, M., Bosco, C., Rosso, P., & Benamara, F. (2020). Multilingual Irony Detection with Dependency Syntax and Neural Models. In *Proceedings of the 28th International Conference on Computational Linguistics*, (pp. 1346–1358). International Committee on Computational Linguistics, Barcelona. https://doi.org/10.18653/v1/2020.coling-main.116. https://aclanthology.org/2020.coling-main.116

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., & Stoyanov, V. (2019). Unsupervised cross-lingual representation learning at scale. CoRR **abs/1911.02116**arXiv:1911.02116

Davidov, D., Tsur, O., & Rappoport, A. (2010). Semi-supervised recognition of sarcasm in twitter and amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, (pp. 107–116).

De Marneffe, M.-C., Manning, C. D., Nivre, J., & Zeman, D. (2021). Universal dependencies. *Computational Linguistics*, *47*(2), 255–308.

De Smedt, T., Daelemans, W. (2012). "vreselijk mooi!" (terribly beautiful): A subjectivity lexicon for Dutch adjectives. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, (pp. 3568–3572). European Language Resources Association (ELRA), Istanbul

de Vries, W., van Cranenburgh, A., Bisazza, A., Caselli, T., Noord, G.V., & Nissim, M. (2019) BERTje: A Dutch BERT Model. arXiv:1912.09582

Delobelle, P., Winters, T., & Berendt, B. (2020). RobBERT: a Dutch RoBERTa-based Language Model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, (pp. 3255–3265). Association for Computational Linguistics, Online. https://doi.org/10.18653/v1/2020.findings-emnlp.292. https://www.aclweb.org/anthology/2020.findings-emnlp.292

del Pilar Salas-Zárate, M., Alor-Hernández, G., Sánchez-Cervantes, J. L., Paredes-Valverde, M. A., García-Alcaraz, J. L., & Valencia-García, R. (2020). Review of English literature on figurative language applied to social networks. *Knowledge and Information Systems*, *62*(6), 2105–2137.

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. CoRR **abs/1810.04805**arXiv:1810.04805

Du, Y., Li, T., Pathan, M. S., Teklehaimanot, H. K., & Yang, Z. (2022). An effective sarcasm detection approach based on sentimental context and individual expression habits. *Cognitive Computation*, *14*(1), 78–90.

Farha, I.A., Oprea, S.V., Wilson, S., & Magdy, W. (2022). Semeval-2022 task 6: isarcasmeval, intended sarcasm detection in english and arabic. In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, (pp. 802–814).

Farías, 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.

Fernando, Z.T., Singh, J., & Anand, A. (2019). A study on the interpretability of neural retrieval models using deepshap. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, (pp. 1005–1008)

Ghosh, A., & Veale, T. (2016). Fracking sarcasm using neural network. In *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, (pp. 161–169)

González-Ibánez, R., Muresan, S., & Wacholder, N. (2011). Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, (pp. 581–586).

Grice, H. P. (1975). Logic and Conversation (pp. 41–58). Brill.

Heafield, K. (2011). KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, (pp. 187–197). Association for Computational Linguistics, Edinburgh, Scotland. https://www.aclweb.org/anthology/W11-2123

Hogenboom, A., Bal, D., Frasincar, F., Bal, M., de Jong, F., & Kaymak, U. (2013). Exploiting emoticons in sentiment analysis. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, (pp. 703–710)

Honnibal, M., & Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. *To Appear, 7*(1), 411–420.

Jijkoun, V., & Hofmann, K. (2009). Generating a non-english subjectivity lexicon: Relations that matter. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, (pp. 398–405).

Joshi, A., Goel, P., Bhattacharyya, P., & Carman, M.J. (2016). Automatic identification of sarcasm target: An introductory approach. CoRR **abs/1610.07091**arXiv:1610.07091

Kralj Novak, P., Smailović, J., Sluban, B., & Mozetič, I. (2015). Sentiment of emojis. *PloS ONE, 10*(12), 0144296.

Kreuz, R., & Caucci, G. (2007). Lexical influences on the perception of sarcasm. In *Proceedings of the Workshop on Computational Approaches to Figurative Language*, (pp. 1–4).

Le, T. T., Fu, W., & Moore, J. H. (2020). Scaling tree-based automated machine learning to biomedical big data with a feature set selector. *Bioinformatics, 36*(1), 250–256.

Liebrecht, C., Kunneman, F., & van Den Bosch, A. (2013). The perfect solution for detecting sarcasm in tweets# not. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, (pp. 29–37). Association for Computational Linguistics, Atlanta, Georgia. Retrieved from https://aclanthology.org/W13-1605

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

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. In: Bengio, Y., LeCun, Y. (eds.) In *Proceedings of the 1st International Conference on Learning Representations, ICLR 2013*, Scottsdale, Arizona

Oostdijk, N., Reynaert, M., Hoste, V., & Schuurman, I. (2013). *The construction of a 500-million-word reference corpus of contemporary written dutch*. Springer, Berlin: Essential Speech and Language Technology for Dutch.

Ordelman, R., de Jong, F., Van Hessen, A., & Hondorp, H. (2007). Twnc: A multifaceted dutch news corpus. *ELRA Newsletter, 12*(3/4), 4–7.

Potamias, R. A., Siolas, G., & Stafylopatis, A.-G. (2020). A transformer-based approach to irony and sarcasm detection. *Neural Computing and Applications, 32*(23), 17309–17320.

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

Riloff, E., Qadir, A., Surve, P., De Silva, L., Gilbert, N., & Huang, R. (2013). Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, (pp. 704–714)

Rohanian, O., Taslimipoor, S., Evans, R., & Mitkov, R. (2018). WLV at SemEval-2018 task 3: Dissecting tweets in search of irony. In *Proceedings of the 12th International Workshop on Semantic Evaluation*, (pp. 553–559). Association for Computational Linguistics, New Orleans, Louisiana. https://doi.org/10.18653/v1/S18-1090. https://aclanthology.org/S18-1090

Sulis, E., Farías, D. I. H., Rosso, P., Patti, V., & Ruffo, G. (2016). Figurative messages and affect in twitter: Differences between# irony,# sarcasm and# not. *Knowledge-Based Systems, 108*, 132–143.

Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*. ICML'17, (pp. 3319–3328). JMLR.org, Sydney

Tsur, O., Davidov, D., & Rappoport, A. (2010).. Icwsm–a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In *Fourth International AAAI Conference on Weblogs and Social Media*.

Van Hee, C. (2017). Can machines sense irony? : exploring automatic irony detection on social media. PhD thesis, Ghent University

Van Hee, C., Lefever, E., & Hoste, V. (2016). Exploring the realization of irony in twitter data. In: Calzolari, N., Choukri, K., Declerck, T., Goggi, S., Grobelnik, M., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., Piperidis, S. (eds.) In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC*, (pp. 1795–1799). European Language Resources Association (ELRA), Paris, France (2016)

Van Hee, C., Lefever, E., & Hoste, V. (2018a). We usually don't like going to the dentist : Using common sense to detect irony on twitter. *Computational Linguistics, 44*(4), 793–832.

Van Hee, C., Lefever, E., & Hoste, V. (2018b). SemEval-2018 task 3: Irony detection in English tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, (pp. 39–50). Association for Computational Linguistics, New Orleans, Louisiana. https://doi.org/10.18653/v1/S18-1005. https://aclanthology.org/S18-1005

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I. (2017). Attention is all you need. In {I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), *Advances in neural information processing systems* Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Wilson, D., & Sperber, D. (2012). Meaning and relevance (CUP: 123–145).

Wu, C., Wu, F., Wu, S., Liu, J., Yuan, Z., & Huang, Y. (2018). THU_NGN at SemEval-2018 task 3: Tweet irony detection with densely connected LSTM and multi-task learning. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, (pp. 51–56). Association for Computational Linguistics, New Orleans, Louisiana. https://doi.org/10.18653/v1/S18-1006. https://aclanthology.org/S18-1006