International Journal of Intelligent Information Technologies
Volume 17 • Issue 4

IronyTR:
Irony Detection in Turkish Informal Texts

Asli Umay Ozturk, Middle East Technical University, Turkey
Yesim Cemek, Middle East Technical University, Turkey
Pinar Karagoz, Middle East Technical University, Turkey
https://orcid.org/0000-0003-1366-8395

ABSTRACT

Irony, which is a way of expression through the use of the opposite, commonly occurs in daily social media posts. Hence, automatic detection of irony is essential to understand the semantics of informal texts more accurately. The literature has several sentiment analysis studies on Turkish texts, but those focusing on irony detection are very few. This paper investigates the effectiveness of a rich set of supervised learning methods varying from traditional to deep neural solutions on Turkish texts. Traditional irony detection methods such as support vector machine (SVM) and tree-based binary classifiers are analyzed on Turkish informal texts. Furthermore, such methods are extended by polarity-based information and graph-based similarity scores as features. Additionally, neural architecture-based solutions including BERT and various LSTM network models are adapted for the problem. Irony detection performance of all the methods are comparatively analyzed on a data set collected within this study, which is larger than the previously used irony detection data sets in Turkish.

KEYWORDS

Classification, Deep Neural Networks, Feature Extraction, Graph Model, Information Extraction, Irony, Sentiment Analysis, Supervised Learning, Text Mining

INTRODUCTION

Together with the rapid growth of online services, there occurred an increasing amount of textual data produced by users every day. There is a need to analyze this huge textual data to understand users' demands, and make deductions for a wide set of applications such as product improvements for e-commerce. In order to meet these needs, sentiment analysis methods, which are useful to extract emotions from textual data, are used (Chakraborty et al., 2020; Yadav et al., 2020).

There is a rich variety of methods to extract emotional information from texts, however these methods lack the ability to analyze irony, especially in Turkish texts. Oxford Dictionary defines irony as *the expression of one's meaning by using language that normally signifies the opposite, typically for humorous or emphatic effect* ("Irony", 2020). From the definition, one can understand why irony is particularly hard to detect: opposition of the meaning is mostly implicit, and automated emotion detection methods lack the understanding of *common sense* that we humans share.

DOI: 10.4018/IJIIT.289965

Irony can be in various forms, which further makes it a complex task to model and detect irony. On this issue, Van Hee and others focus on two irony types: *situational irony* and *irony by means of a polar clash.* Any ironic text that does not fit in to these two categories are referred to as *other textual irony*. They define situational irony as the type of irony where the text is a written description of an ironic situation, and claim that this type of irony is harder to detect and model since it needs an understanding of the context (Van Hee et al., 2016a). On the other hand, Carvalho and others focus on this type of irony in their study conducted in Portuguese (Carvalho et al., 2020). Following is a good example of situational irony:

COVID-19 toplantısı, COVID-19 önlemleri kapsamında iptal edildi.
COVID-19 meeting is cancelled due to COVID-19 precautions. (transl.)

As the second type, Van Hee and others define irony by means of a polar clash as the type of irony where two opposite sentiments can be extracted from the same text (Van Hee et al., 2016a). Since this type of irony can be modeled by sentiment scoring of the tokens, several studies conducted on English use sentiment scores of words to improve the performance of their models (Ahmed et al., 2018; Van Hee et al., 2016b; Xu et al., 2015). Following sentence clearly illustrates irony by means of a polar clash:

Sabrımı denemeni *çok* **seviyorum***!*
I just **love** *when you* **test my patience***! (transl.)*

Irony detection on text can be of practical use in a variety of applications that involve text analysis. On its own, irony detection facilitates revealing the language use and language style of the author. Such analysis provides useful input for psychological studies on personality or mental health (Bruntsch & Ruch, 2017). One of the most prominent domains that can get help from irony detection is marketing and customer relations management (CRM). Sentiment analysis is widely used in social media analysis for CRM in order to understand the customer opinions. However social network postings on the products including ironic expressions distort the correctness of automated sentiment analysis. Applying irony detection together with sentiment analysis can improve such analysis to capture user's opinion accurately (Ravi et al., 2017; Alt & Reinhold, 2020). Another direction for employing irony detection in CRM is through personality analysis. Understanding the personality traits of customers has an important role in marketing to determine personalized offerings (Ramadhanti et al., 2020). There are recent studies to perform personality traits analysis on texts and social media messages written by users in an automated manner (Tutaysalgir et al., 2019). Irony detection can augment such studies to improve personality traits detection.

In the literature, there are several different approaches devised for the problem of irony detection. Most of the studies approach the problem as a binary classification task, and extract different features from the text to improve the detection model. For the binary classification problem, both traditional supervised learning methods and neural network based solutions are used. Most of such studies are conducted on English texts (Ahmed et al., 2018; Baloglu et al., 2019; Barbieri et al., 2014; Buschmeier et al., 2014; Pamungkas & Patti, 2018; Van Hee et al., 2016b; Wu et al., 2018; Zhang et al., 2019), whereas there are a few recent studies conducted on other languages (Carvalho et al., 2020; Frenda & Patti, 2019; Ghanem et al., 2020; Xiang et al., 2020). When it comes to Turkish, there exists only a few works conducted on limited data sets with limited features (Dulger, 2018; Taslioglu & Karagoz, 2017).

The aim of this study is to investigate the irony detection of a variety of supervised learning models that have not yet been applied on Turkish texts in the earlier studies. Furthermore, we conduct the analysis on a recently collected data set in Turkish, which is larger than those used previously. For this goal, both neural network based models and traditional supervised learning methods are

utilized, and a wide set of features aiming to exploit different types of irony are used. Since we focus on identifying written expressions of any type of irony, considering that polar clash is a commonly used irony type, we exploit polarity scores to detect occurrences of polar clash and to improve the constructed irony detection models. Also, similar to the approach by (Ahmed et al., 2018), graph representations are used to extract graph similarity based features. In our work, the feature set is divided into subsets and they are used incrementally to detect the effect of different features on the irony detection performance.

As a summary, the goal of this study is to analyze and elaborate on the irony detection performance on Turkish texts for those methods that have been applied on English and other languages in the literature. Furthermore, we use a comprehensive irony data set in Turkish. The contributions of this work can be listed as follows:

- We investigate the detection performance of the traditional supervised learning methods for our problem. In addition to conventional features extracted from the text, we exploit graph-based and polarity-based features in order to improve irony detection performance. We analyze the performance of different pipelines through a genetic-algorithm based optimization solution.
- In addition to traditional methods, we adapt neural models for our problem. For BERT, in addition to the fully pretrained model, we analyze the performance of training level variations. For LSTM, bi-LSTM and CNN-LSTM, in addition to training with text content, we extend the models with additional features extracted from the text.
- The experiments are conducted on an irony data set in Turkish, which is larger than the earlier collections used in the literature. It is collected and annotated within the scope of this study. This data set is open for research purposes and can be accessed from our Github page (https://github.com/teghub/IronyTR).

The rest of the paper is organized as follows. Firstly, we present the background in irony detection in terms of methods used in the literature. Then the methods and data processing and model construction pipeline used in the study are presented. It is followed by description of the experiments and discussion of the results. The paper is concluded with an overview on the conducted study and obtained results, as well as future research directions.

BACKGROUND

In the literature, irony detection is considered as a text classification problem, and is basically modeled as a binary classification including the classes of irony and non-irony. The previous studies conducted on English texts use supervised learning methods for binary classification (Baloglu et al., 2019; Barbieri et al., 2014; Buschmeier et al., 2014; Pamungkas & Patti, 2018; Van Hee et al., 2016b).

In their study, Buschmeier and others compare the performance of different feature set and classification method combinations. Using supervised classification methods such as Support Vector Machine (SVM), Random Forest (RF), Logistic Regression, Decision Tree (DT) and Naive Bayes (NB) with several syntactic and lexical features, they work on a product review data set. The authors also utilize product star rating included in the data set, and claim that using the star-rating meta-data results in a competitive performance. However, such meta-data is not generally available in most of the text-based data (Buschmeier et al., 2014). Barbieri and others consider six classes in their study, namely Sarcasm, Education, Humour, Irony, Politics and Newspaper. Using a tree-based classifier, they aim to eliminate the use of patterns of words as features and they propose seven sets of lexical features such as Frequency (gap between rare and common words), Intensity (intensity of adverbs and adjectives), Synonyms (common vs. rare synonyms use) that will help to detect sarcasm by the inner structure of the expressions (Barbieri et al., 2014). In their study, Van Hee and others use an SVM based model to first classify tweets into two categories as ironic and non-ironic, then they further

classify ironic tweets into three subcategories as situational irony, irony by means of polar clash and other verbal irony. A comprehensive feature set with basic lexical, syntactic, semantic and sentimental features are used to train the model (Van Hee et al., 2016b). Pamungkas and Patti use a similar system with a limited feature set, additionally they exploit the sentiment analysis of *emoji*s to capture more information about the text (Pamungkas & Patti, 2018). Baloglu and others study several supervised machine learning algorithms including K-Nearest Neighbours (k-NN), and Decision Tree Learning on a similar set of features to analyze their detection performance (Baloglu et al., 2019). Ahmed and others utilize graph representations of sentences to extract new features from textual data and use supervised learning to train their model on these features (Ahmed et al., 2018).

Previously mentioned studies all discuss and report about the difficulty of detecting situational irony. On this issue, Carvalho and others use a Portuguese data set of farcical news headlines to explore different rhetorical devices that are used to construct irony. They model these rhetorical devices by using general lexical and syntactic features, as well as a measure for degree of predictability of a situation. Using this measure, the authors model out-of-domain contrast, which is important for understanding situational irony. They conclude that sentiment features are not enough to capture situational irony and using a measure for out-of-domain contrast improves the performance (Carvalho et al., 2020).

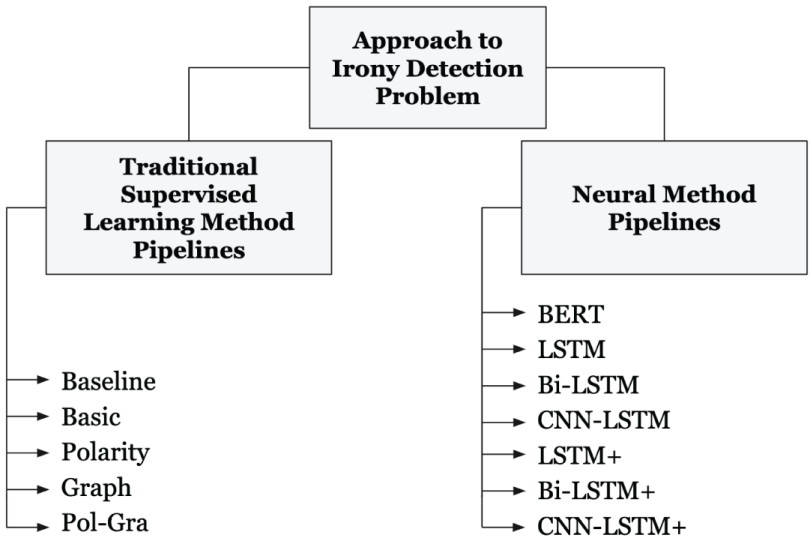
Another approach to binary classification problem of irony is by neural-based methods. Wu and others propose a system based on a densely connected Long-Short Term Memory (LSTM) with multi-task learning strategy. They also use additional features such as Part of Speech (PoS) tags, which indicate the role of a word in a sentence, sentiment features and sentence embeddings (Wu et al., 2018). Zhang and others brought a different perspective to the existing solutions and they integrated the sentiment information obtained from external sources to supervised learning on irony labeled text using transfer learning. By this, they aim to increase the ability of attention-based model on detecting context incongruity. Their results show that transferred sentiment increases the models' ability to detect implicit and explicit context incongruity and their three proposed sentiment attention mechanisms outperform the baselines including several popular neural network models for irony detection on Twitter (Zhang et al., 2019).

Although there exist several studies conducted on languages other than English (Carvalho et al., 2020; Frenda & Patti, 2019; Ghanem et al., 2020; Xiang et al., 2020) there are very few studies on irony detection in Turkish. Dulger studies binary classification of irony using SVM, k-NN, NB, RF as well as Logistic Regression and Multilayer Perceptron on a balanced Turkish data set of 144 instances with a limited set of features (Dulger, 2018). Taslioglu and Karagoz study the binary classification of irony on a larger and balanced Turkish data set of 194 instances, using a limited set of features with SVM, k-NN, NB and RF classifiers. Similar to Van Hee and others (Van Hee et al., 2016b), they also include polarity score based sentimental features in their data set (Taslioglu & Karagoz, 2017). Another study conducted on Turkish, which is a preliminary version of this study, compares the performances of SVM, NB and LSTM based classifiers and a language model (Devlin et al., 2018) BERT (Bidirectional Encoder Representations from Transformers) on a balanced Turkish data set of 220 instances, which is larger than the previously used data sets (Cemek et al., 2020).

METHODS

Our work focuses on the binary classification problem of textual data to two categories, *ironic* and *non-ironic*. It is a subset of sentiment analysis problem, and is considered to be more difficult than sentiment analysis. One reason is that, in some cases, detecting the existence of irony in a sentence can be challenging even for us humans. It is a complex concept that cannot be grasped solely by comparing the sentiment scores of words in a sentence. Hence, we investigate the irony detection performance of supervised learning methods taking multiple aspects of irony into consideration.

Figure 1. Methods and approaches



Due to limitations on the data sets used in the previous studies, we collected an irony detection data set in Turkish within the scope of this study. The performance of neural approaches as well as polarity score and graph-based features are investigated for the irony detection problem in Turkish. Also, known methods that have been used for the problem of irony detection in English are applied to Turkish to investigate their performance on Turkish language. For comparison, *Bag of Words (BoW)* representation is used as the only feature in the baseline classifier.

For different methods, different preprocessing and feature selection pipelines are used. In Figure 1, we summarize the methods used in this work. In the rest of this section, we describe the data collection, preprocessing and feature extraction, pipeline optimization used for the traditional methods and the neural models used for the problem.

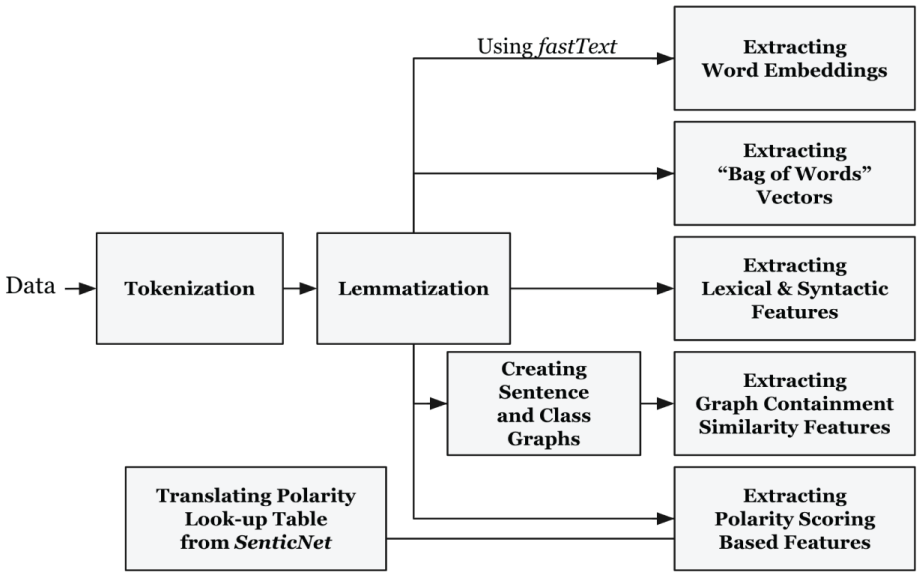
Data Collection and Data Set

The data set used in this study is collected from Twitter and other microblogs/social media platforms. When collecting data, hashtags and other signifiers of the topical events are used to search for possible instances via the API's of such platforms. Annotation of the collected data is performed by a group of 7 native Turkish speakers, and the ground truth is set through qualified majority voting. Hence, instances that are labeled with a close-vote (4/7 in favor for a label) are excluded from the dataset to prevent any ambiguity. Final data set contains 600 instances, with 300 ironic and 300 non-ironic samples. The basic statistics on the data set are given in Table 1.

Table 1. Data set statistics

Class Labels	Number of instances	Average token length per instance	Maximum token length per instance	Minimum token length per instance	Number of instances including emoticons	Number of instances including !, ?, (!), (?), …, ""
Ironic	300	10	25	3	26	94
Non-ironic	300	9	28	3	7	53

Figure 2. Preprocessing and feature extraction pipeline



Preprocessing and Feature Extraction

Due to the nature of the methods employed, we applied various feature extraction pipelines for different methods. Feature extraction phase starts with preprocessing of the text. The feature extraction and preprocessing pipeline is summarized in Figure 2. For preprocessing, each sentence is tokenized by words, punctuation marks and emojis/emoticons. All letters are converted to lowercase. An example preprocessing is as follows.

"*Sınava geç kaldım, aferin bana!*" is transformed into the following tokens: "*sınav geç kalmak, aferin ben !*" where tokens are "*sınav*", "*geç*", "*kalmak*", ",", "*aferin*", "*ben*", "*!*".

After preprocessing, the following basic syntactic and lexical features, which are also used in several previous studies (Dulger, 2018; Frenda & Patti, 2019; Taslioglu & Karagoz, 2017; Van Hee et al., 2016b), are extracted:

- **Word Count:** A floating point value indicating the ratio of words to all tokens.
- **Interjections:** A binary value indicating if there is any interjection words ("bravo", "oley (transl. hurray)" etc.) in the text.
- **Boosters:** A binary value indicating if any booster words ("asla (transl. never)", "mutlaka (transl. of course") etc.) exist.
- **Repetition:** A binary value indicating if there is any repeated tokens in the sentence.
- **Capitalization:** A binary value indicating if there is any capitalized words in the sentence. (This feature is extracted before converting every letter to lowercase.)
- **Emoji/Emoticons and Punctuation Marks (Exclamation, Question, Ellipsis, Quotation, Bracketed Exclamation, Bracketed Question Marks):** For each of these tokens, two features are extracted. One of them is a binary value indicating if the token exists in the text. The other one is a floating point value indicating the ratio of the count of token to the count of all tokens. Therefore, on the total, 14 features are extracted for these tokens.
- **All Punctuation Marks:** A floating point value indicating the ratio of punctuation mark count to all token count.
- **Bag of Words:** A vector of the size of the whole corpus, where count of each token in the sentence is shown with a normalized floating point value.

As seen in the above feature list, for some tokens such as several punctuation marks and emoji/emoticons, two features are extracted. This is due to that they capture different semantics (one of them is about existence of the token and the other is about the quantity), and hence may have different impact on the irony detection performance of a model.

In the previous irony detection studies, polarity scores of words are known to be used in order to detect positive or negative orientation in sentiment (Ahmed et al., 2018; Dulger, 2018; Taslioglu & Karagoz, 2017; Van Hee et al., 2016b; Xu et al., 2015). Since there is no publicly available polarity score look-up library for Turkish, in the studies conducted on Turkish, generally, the English libraries are translated manually (Vural et al., 2013). In this work, we manually translated the words in our data set and created our own look-up table by using *SenticNet* (Cambria et al., 2020). This look-up table includes scores between -1.0 and 1.0 for the words in our collection. A sample polarity scoring for the sentence "*Sınava geç kaldım, aferin bana!*" is shown in Figure 3.

After constructing the polarity scores look-up table, we extracted the following features:

- **Average Polarity:** For average polarity, we have three features of floating point value. The first one keeps the positive average polarity, the second one has the negative average polarity and the third one is total average polarity. Each of these average values are calculated by using the number of sentimental tokens in the text.
- **Minimum Polarity:** A floating point value indicating the minimum polarity score in the sentence.
- **Maximum Polarity:** A floating point value indicating the maximum polarity score in the sentence.
- **Maximum Polarity Difference:** A floating point value indicating the difference of minimum and maximum polarities, scaled in [0,1].
- **Positive and Negative Sum Difference:** A floating point value indicating the difference of positive sum and negative sum, scaled in [0, 1].
- **Polarity Contrast:** A binary value indicating the existence of both positive and negative polarity scores in the sentence.

One of the enhancements we use in our study is exploiting graphs to discover new features that may contain hidden relationships within the text. The basic idea here is to create class graphs for ironic and non-ironic data in order to extract several graph similarity scores as features.

Creating Sentence Graph

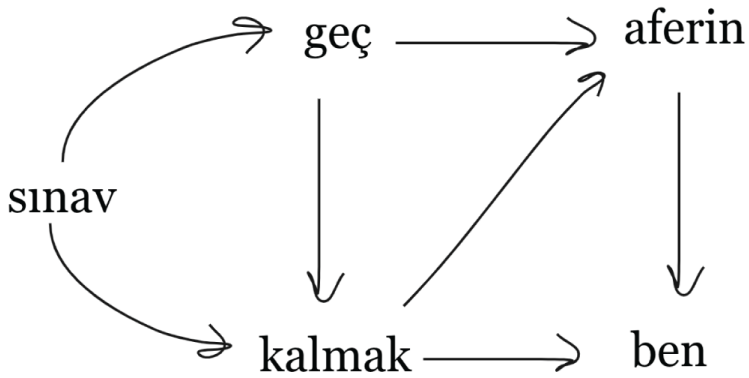
Using the method described in the study by Ahmed and others, a vicinity window of 3 is used for creating a graph for each sentence in the data set, where each token has a directed edge to next two tokens following itself. Only word and emoji/emoticon tokens are used to create graphs (Ahmed et al., 2018). As an example, the graph constructed for the sentence "*Sınava geç kaldım, aferin bana!*" is shown in Figure 4.

Creating Class Graphs

A class graph is created by taking the union of the sentence graphs of the sentences in that class. Hence, we have two class graphs, one for irony and the other for non-irony class. Using the graphs, the following features are extracted:

Figure 3. Polarity scores of the words in a sentence

Figure 4. Sample sentence graph constructed for the sentence "Sınava geç kaldım, aferin bana!" with vicinity window of 2



- **Containment Similarity Score:** For containment similarities, we keep two floating point values each one representing the containment similarity score of a sentence graph to ironic and non-ironic class graphs. Containment similarity is calculated as given in Equation 1, where S is the sentence graph and C is the class graph. Here, the size of a graph and the size of the intersection of two graphs are calculated with respect to the number of edges:

$$\frac{\left|S \cap C\right|}{\min\left(\left|S\right|,\left|C\right|\right)} \tag{1}$$

Additionally, in order to use in the neural pipelines, word embeddings are extracted from the lemmatized data:

- **Word Embeddings:** Word embeddings of the tokens are obtained by using the pre-trained embedding model of *fastText* (Bojanowski et al., 2017) trained on Turkish Wikipedia texts.

Note that not all features are used in all of the methods. The use of the features in the learning pipelines are summarized in Table 2.

Optimizing the Pipeline for Traditional Supervised Learning Methods

Irony detection, as a supervised learning task, involves a pipeline using traditional classification methods such as SVM, NB and DT classifiers. Selecting the best fitting method and optimizing the parameters for the best performing setting requires a considerable effort. To facilitate this phase, we used an automated machine learning tool, *TPOT* (Olson et al., 2016). Given your training data as a feature vector, TPOT uses genetic programming to optimize the machine learning pipeline for the given task, according to the metric defined. In this study, TPOT is used with SVM, Multinomial NB and DT classifiers for the optimization of traditional supervised learning methods. It should be noted that, in our study, TPOT is only used for tuning parameters of the classification methods. We exclude the other optimization functionalities, especially for feature selection since we aim to analyze the effect of the features explicitly. The details of the pipelines and methods used in the analysis are given in the Experiments & Results section.

Table 2. Features used in pipelines

Features	Baseline	Basic	Polarity	Graph	Pol-Gra	LSTM	Bi-LSTM	CNN-LSTM	LSTM+	Bi-LSTM+	CNN-LSTM+
BoW	x	x	x	x	x						
Word Cnt.		x	x	x	x				x	x	x
Emo.		x	x	x	x				x	x	x
Intjct.		x	x	x	x				x	x	x
Boost.		x	x	x	x				x	x	x
Repet.		x	x	x	x				x	x	x
Caps		x	x	x	x				x	x	x
Excl.		x	x	x	x				x	x	x
Q.		x	x	x	x				x	x	x
Ellp.		x	x	x	x				x	x	x
Quote.		x	x	x	x				x	x	x
Brckt. Excl.		x	x	x	x				x	x	x
Brckt. Q.		x	x	x	x				x	x	x
All Punct.		x	x	x	x				x	x	x
Avg Pol.			x		x						
Min Pol.			x		x						
Max Pol.			x		x						
Max Dif.			x		x						
Pos-Neg D.			x		x						
Pol. Cont.			x		x						
Cont. Sim.				x	x						
Word Emb.						x	x	x	x	x	x

Neural Network Based Methods

As the neural network based solutions, we used BERT and several variations of LSTM neural model with a pre-trained word embeddings by fastText on Turkish Wikipedia texts. Using the corresponding embedding vectors for each word in our data set, an embedding matrix is created and used to form the embedding layer of the neural models. For each method explained below, the same embedding layer is used and trained further with the rest of the network. Also, for LSTM based models, we experimented with additional features to create new models, which are explained in detail in the Experiments & Results section:

- **LSTM:** LSTM is a well-known recurrent neural network (RNN) architecture. Different from the standard artificial neural network architectures, it contains feedback connections which provides long-term memory to recurrent neural networks. With this feature, it is ideal for working on time series and textual data.
- **Bi-LSTM:** Bidirectional Long-Short Term Memory networks (Bi-LSTM) run the given inputs twice, from past to future and from future to past. In this way, unlike the regular LSTM networks, it not only uses the information from the past, but also combines it with the information from the future and makes it possible to access this at any time. By adding an embedding layer to the established architecture, vector representations of the texts can be obtained and used in the model training process.

- **CNN-LSTM:** It is an LSTM architecture specifically designed for sequence prediction problems with spatial inputs, like images or videos. It contains convolutional layers, used in Convolutional Neural Networks (CNN), for feature extraction on input data combined with LSTMs for text classification.
- **BERT:** *BERT* is a masked language model (Devlin et al., 2018) developed by Google Research. Its main difference from the existing language models is that it is pre-trained in a bidirectional manner, allowing the model to capture the relations of a token with both its previous and subsequent tokens. In this study, *BERT Base Multilingual Cased* pre-trained model, which is trained on cased text in the top 104 languages with the largest Wikipedia content, is fine-tuned for binary classification of textual data.

EXPERIMENTS AND RESULTS

Within the scope of our study, we investigate and compare several methods (as shown in Figure 1). Features used in each pipeline are presented in Table 2. All experiments are performed by using the same data set under 10-fold cross validation.

Traditional Supervised Learning Methods

We analyzed the performance of 5 main pipelines utilizing traditional supervised learning methods, where each pipeline mainly differs by the set of features used in classification. The pipelines are described in detail below. For each pipeline, Multinomial NB, SVM and DT classifiers are optimized and their results are compared in Table 3:

- **Baseline Pipeline:** Bag of Words vectors contain the minimal information we can get from a sentence. Hence, to have a baseline method where the performances of the other methods can be compared against, *Baseline* pipeline is created by only using BoW vectors. It should be noted that neural pipelines are also compared against this *Baseline* pipeline.
- **Basic Pipeline:** As seen in Table 2, for the basic features pipeline, only the lexical and syntactic features are included for the feature set, on top of the BoW vectors.
- **Polarity Pipeline:** Polarity scores can contain information to facilitate irony detection, hence polarity based features are added to the features used in the *Basic* pipeline (as shown in Table 2) to create the feature set for the *Polarity* pipeline.
- **Graph Pipeline:** Graph similarities can also contain important or hidden information. To utilize graph similarities, containment similarity score features are added to the features used in the *Basic* pipeline (as shown in Table 2) to create the feature set for the *Graph* pipeline.
- **Pol-Gra Pipeline:** Finally, both graph and polarity based features are added on top of the features used in the *Basic* pipeline to create the *Pol-Gra* pipeline.

Neural Network Based Pipeline

In the pipelines, *LSTM*, *Bi-LSTM* and *CNN-LSTM*, word embeddings are used as the input. On the other hand, for the pipelines of pipelines, *LSTM+*, *Bi-LSTM+* and *CNN-LSTM+*, in addition to the word embeddings, basic features are added before the dense layers. The architectures of these two groups of networks are the same and can be seen in Figure 5. The detection performance of the neural pipelines are compared in Table 5. In the *BERT* pipeline, none of the extracted features are used, tokenized data is fed directly to the BERT model. Hence, *BERT* pipeline is not shown in Table 2:

- **LSTM:** For the *LSTM* model, the experiments are conducted with different combinations of settings. The tested settings are as follows: epoch numbers 3 and 5, the number of LSTM nodes

Figure 5. LSTM based architectures

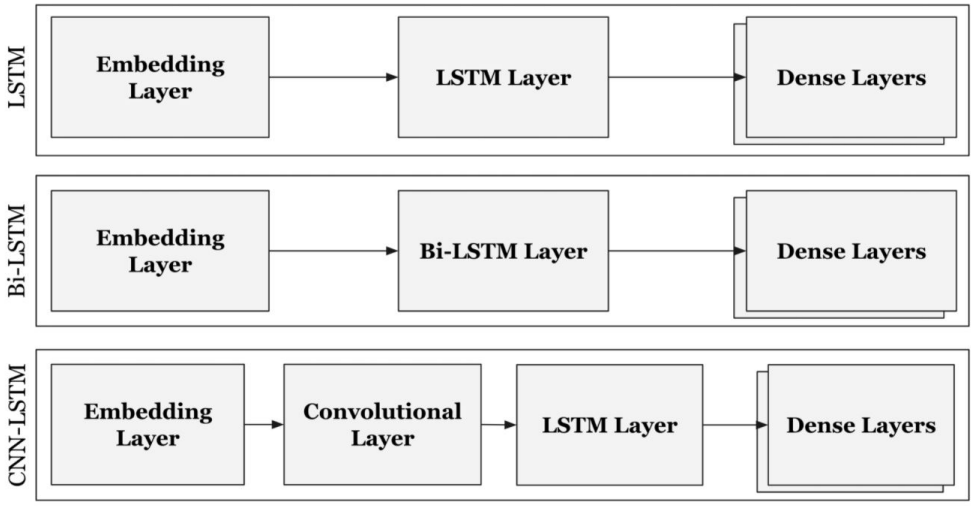


Table 3. Comparison of traditional methods

Pipeline/Method		Accuracy	Precision	Recall	F1-score
Baseline	SVM	48.17%	29.64%	56.33%	38.84%
	Multinomial NB	48.17%	39.54%	56.32%	46.46%
	Decision Tree	52.17%	52.19%	35.17%	42.02%
Basic	SVM	53.50%	60.01%	66.34%	63.02%
	Multinomial NB	55.33%	58.83%	68.44%	63.27%
	Decision Tree	56.67%	64.58%	39.74%	49.20%
Polarity	SVM	**63.33%**	64.57%	58.22%	61.23%
	Multinomial NB	55.83%	58.23%	74.35%	65.31%
	Decision Tree	53.50%	58.64%	56.56%	57.58%
Graph	SVM	53.50%	60.01%	66.34%	63.02%
	Multinomial NB	58.17%	61.40%	67.90%	64.49%
	Decision Tree	54.83%	58.86%	62.43%	60.59%
Pol-Gra	SVM	**63.33%**	**65.53%**	57.10%	61.03%
	Multinomial NB	55.67%	58.11%	**76.12%**	**65.91%**
	Decision Tree	55.50%	56.00%	50.40%	53.50%

64, 128, 256 and the number of dense layers 2 and 3. Different activation functions for the dense layers are also tested, such as sigmoid and ReLU. The best performance is obtained under the configuration including 5 epochs, with 128 LSTM nodes, 3 dense layers with 100, 10 and 1 outputs respectively and ReLU activation function.

- **Bi-LSTM:** The *Bi-LSTM* model is also tested with different combinations of settings. The following configurations have been tested: epoch numbers 3 and 5, the number of Bi-LSTM nodes 64, 128, 256 and the number of dense layers 2 and 3. Sigmoid and ReLU activation functions are compared to see the effect of the activation function. The best performance is obtained with

3 epochs, with 128 Bi-LSTM nodes, 2 dense layers with 100 and 1 outputs respectively and ReLU activation function.

- **CNN-LSTM:** The configurations used for the experiments of the *CNN-LSTM* network are, epoch numbers 3 and 5, the number of kernels 64, 128, and 256, the kernel sizes 4, 6 and 8 and the number of dense layers 2 and 3. The number of nodes in the LSTM layer is adjusted according to the selected kernel size and number of kernels. Sigmoid and ReLU activation functions are tested. The best result is obtained under the following configuration: with 5 epochs, using 128 kernels with size 6, 2 dense layers having 100 and 1 outputs with ReLU activation function.
- **LSTM+:** The experiment settings used for *LSTM+* model is the same with the LSTM model. The best performance for this pipeline is obtained with 3 epochs, with 128 LSTM nodes, 2 dense layers having 100 and 1 outputs, respectively, with ReLU activation function.
- **Bi-LSTM+:** The *Bi-LSTM+* model is experimented with the same settings as the *Bi-LSTM* model. The best performance for this pipeline is obtained with 5 epochs, with 64 Bi-LSTM nodes, 3 dense layers with 100, 10 and 1 outputs, respectively, and ReLU activation function.
- **CNN-LSTM+:** The *CNN-LSTM+* model is also experimented with the same settings as the CNN-LSTM model. The best performance is obtained with 5 epochs, using 64 kernels with size 6, 2 dense layers having 100 and 1 outputs, with ReLU activation function.
- **BERT:** Using an open-source implementation (Rajapakse, n.d.) with limited changes for weight freeze implementation, we constructed a 12-layered BERT model. Hyperparameter settings of epoch numbers 5, learning rate 0.00004, batch size 16 are kept the same while analyzing different weight freeze settings. All results are evaluated by 10-fold cross validation and are shown in Table 4, where *Layers* indicate the last number of layers that are not frozen, in other words, layers where the parameters are trained.

Evaluation Metrics

The experiments are conducted under 10-fold cross validation using the following 4 metrics: accuracy, precision, recall and F1-score. These metrics are calculated by using their conventional equations involving the number of *true positive*, *true negative*, *false positive* and *false negative* instances. For cases where there is no prediction generated for a given class or there is no instance for a class in the test set, precision and recall calculations may result in division by zero. To handle this situation, a method proposed in *GERBIL* project is used (Roeder, 2015). According to their solution, when true positives, false positives and false negatives are 0, the precision, recall and F1-measure values are given as 1. When true positives are 0 and one of the other measures is larger than 0, the precision, recall and F1-measure are considered to be 0.

Traditional Pipeline Results

To select the best classifier for each pipeline, three classifiers are optimized with the help of TPOT. For each pipeline, the highest score by column is written in bold, the best performing

Table 4. Comparison of BERT with different trained layers

Layers	Accuracy	Precision	Recall	F1-score
11	68.06%	68.92%	63.67%	66.19%
7	64.83%	66.46%	60.17%	63.16%
6	**69.00%**	**71.34%**	**65.75%**	**68.43%**
5	66.50%	67.23%	61.64%	64.31%
1	63.50%	65.21%	59.76%	62.37%

pipeline is selected by the highest F1-score, and the selected pipelines are also written in bold in Table 3. In the table, it is seen that incrementally adding more features results in a trend of increasing performance for each classification method. For the comparison of polarity-based and graph-based features separately, we can see that polarity scores improves the performance of systems more. We can see an accuracy boost in SVM when polarity scores are included. But since the best performing classifiers for each pipeline are selected by F1-scores, for all pipelines, Multinomial Naïve Bayes classifier is preferable.

Neural Pipeline Results

For the analysis of neural models, initially the BERT settings with different training layers are evaluated. In the pre-trained model of the BERT, only the dense layer is trained. The irony detection performance for various settings are given in Table 4. As seen in the results, the best performance is obtained when the last 6 layers are trained. Hence this pipeline is used for comparison with the other models as given in Table 6. For LSTM based methods, as it can be seen in Table 5, the addition of basic features did not change the accuracy scores significantly. For some of the models they improved the performance in terms of F1-score, whereas for the others, the performance decreased. On the overall, CNN-LSTM and CNN-LSTM+ have lower performance than the others. The highest score in terms of F1-score is obtained by LSTM+.

Discussion

For each of the pipelines, the best performing setting in terms of F1-score is included in Table 6. For each metric (column), top-3 performances are written in bold. We can summarize the most prominent result as follows:

- Since BoW vectors contain limited information about each sentence, BoW pipeline is used as the *Baseline* pipeline for comparison, and it has a lower performance. The other traditional pipelines are built on top of the BoW features, by incrementally adding extra features. As expected, incremental addition of features in traditional supervised learning method based pipelines enhanced the performance of the classification task. However, the performance of neural models are not affected significantly by inclusion of basic features to the word embeddings. We believe that this is due to the nature of neural-network based learning process, which actually improves with the quantity of data.
- For LSTM based models, the performance fall below the traditional models. Data set used in this study is relatively small for neural methods to perform at their fullest. But when it comes to BERT pipeline, due to its usage of a multilingual pretrained model for transfer learning, it provides the best performance in terms of accuracy, precision and F1-score.

Table 5. Comparison of LSTM based methods

Pipeline	Accuracy	Precision	Recall	F1-score
LSTM	51.33%	**55.09%**	52.73%	53.88%
LSTM+	50.50%	49.88%	**64.57%**	**56.28%**
Bi-LSTM	50.16%	51.44%	62.07%	56.26%
Bi-LSTM+	**51.66%**	52.61%	56.74%	54.60%
CNN-LSTM	50.33%	50.33%	45.73%	47.92%
CNN-LSTM+	50.33%	46.73%	45.59%	46.15%

Table 6. Comparison of methods

Pipeline	Accuracy	Precision	Recall	F1-score
Baseline	48.17%	39.54%	56.32%	46.46%
Basic	55.33%	**58.83%**	**68.44%**	63.27%
Polarity	**55.83%**	58.23%	**74.35%**	**65.31%**
Graph	**58.17%**	**61.40%**	67.90%	64.49%
Pol-Gra	55.67%	58.11%	**76.12%**	**65.91%**
LSTM	51.33%	55.09%	52.73%	53.88%
LSTM+	50.50%	49.88%	64.57%	56.28%
Bi-LSTM	50.16%	51.44%	62.07%	56.26%
Bi-LSTM+	51.66%	52.61%	56.74%	54.60%
CNN-LSTM	50.33%	50.33%	45.73%	47.92%
CNN-LSTM+	50.33%	46.73%	45.59%	46.15%
BERT	**69.00%**	**71.34%**	65.75%	**68.43%**

- Among the pipelines of traditional classifiers, *Pol-Gra* pipeline, in which all the extracted features (except word embeddings) are included, perform the best. This pipeline provides the highest recall score and immediately follows the BERT in terms of F1-score. In general, traditional pipelines provide higher recall results in comparison to the BERT.
- The results obtained in our analysis are mostly comparable with those obtained in irony detection studies in English. In their study, Ahmed et al. reported irony detection performance of 59.55% F-measure on SemEval-2018 data set by extreme gradient boosting classifier (Ahmed et al., 2018). On the same data set, the performance is further improved to 63% with SVM. Wu et al. used deep neural architectures and obtained 70.5% F1-score by using a LSTM based model (Wu et al., 2018). In our analysis, we obtained the highest detection performance with a deep neural architecture, however LSTM based models' performance remained limited. This is an expected result due to the difference in size of our data set and SemEval-2018 data set (see Table 7).
- Irony detection accuracy results reported for other languages vary. This is possibly due to the differences in data set characteristics as well as the nature of the languages. Carvalho et al. reported about 72% F1-score with SVM on a data set in Portugese (Carvalho et al., 2020). On a collection of tweets and news comments in Spanish, about 62% F1-score is reported by using SVM (Frenda & Patti, 2019). Ghanem et al. reported 68% F1-score on their Arabic data set with RF and 80% with CNN (Ghanem et al., 2020). In the same study, F1-score results obtained on for French data set are 61% by using RF, and 73.5% by using CNN. In another study (Zhang et al., 2019), by using a BERT model, 57% F1-score is reported for irony detection on a Chinese data set.
- Irony detection accuracy results reported in the previous studies on Turkish texts are higher than ours. Dulger reported to obtain reported 88% F1-score with Multi Layer Perceptron (MLP) (Dulger, 2018), and Taslioglu & Karagoz reported 95% with k-NN (Taslioglu & Karagoz, 2017). However data set sizes used in these two studies are smaller than the data set used in our study (see Table 8). Since these data sets are not publicly available, comparative analysis on them was not possible. Since the techniques used in both of these previous studies are traditional supervised learning methods and the features extracted are similar to those in our *Basic* pipeline, the difference in the detection accuracy possibly due to data sets.

CONCLUSION

In this paper, we investigate the performance of traditional supervised learning based methods and neural network-based solutions for irony detection on Turkish informal texts. The analysis is conducted on a data set of 600 instances, which is larger than the data used in the earlier studies. We also analyze the effects of polarity score and graph-based features on recognizing irony. The experiments show that traditional methods generally perform better for this data set and the neural models fall behind since the size of the data set is relatively small. It is also observed that polarity score and graph based features improve the performance of traditional classifiers. Only BERT, an up-and-coming transfer learning language model bypasses all pipelines and gives promising results even with a relatively small data set for neural methods.

One limitation of our study is about the size of the data set used in the analysis. The data set includes 600 sentences on the total with balanced number of instances. Due to the difficulties stemming from the nature of the problem, annotation constitutes an important barrier for generating ground truth data sets. Hence irony detection data sets are on the overall limited in size compared to other text analysis tasks such as sentiment analysis. In Table 7, the characteristics of English irony detection data sets used in the previous studies are summarized. Among these data sets, Barbieri 2014 and Ptáček 2014 are considerably larger in size. However, these data sets are automatically annotated by using hash tags, and no further manual annotation is reported. In Table 8, irony detection data sets in other languages, including Turkish, in the literature are given. Compared to Dulger 2018 and Taslioglu 2017, the size of the data set used in our study is considerably extended, especially for the irony class. For the data sets in Portugese (Carvalho 2020), and in Spanish (IroSvA-Spain, IroSvA-Mexico, IroSvA-Cuba), the sizes of the collections are smaller than SemEval 2018. For the data collection in Chinese (Ciron), the size of ironic samples is limited compared to non-irony class. In the data collection and annotation phase, we put our best effort to generate the data set, and we aim to extend it gradually. Yet with the current size, we believe it was effective to present the trend and potential of the methods for the challenged task.

For the future work, more advanced neural architectures and different classifiers such as k-NN or RF classifier may be utilized. As another research direction, the effect of further graph-based features such as maximum common (induced/edge) sub-graph similarity or token and character N-grams can be investigated for performance improvement. As additional features, PoS taggings of the tokens in sentences can be utilized as well.

Table 7. Summary of irony detection data sets in English

Data set	Size of Ironic Samples	Size of Non-ironic Samples	Total Size	Studies used in	Collection Mechanism
SemEval 2018	2.396	2.396	4.792	(Ahmed et al., 2018), (Pamungkas & Patti, 2018), (Wu et al., 2018)	Twitter, Manual Annotation
Barbieri 2014	10.000	10.000	20.000	(Barbieri et al., 2014)	Twitter & news, Automated Annotation
Filatova 2012	437	817	1.254	(Buschmeier et al., 2014)	Amazon reviews, Manual Annotation
Ptáček 2014	5.602	5.623	11.225	(Ghanem et al., 2020)	Twitter, Automated Annotation

Table 8. Summary of irony detection data sets in Turkish and other languages (* Data set annotation procedure is not given in full detail in (Frenda & Patti, 2019) however since the data set is constructed for a text analysis challenge, manual annotation looks probable.)

Data set	Size of Ironic Samples	Size of Non-ironic Sample	Total Size	Studies used in	Collection Mechanism
Dulger 2018	72	72	144	(Dulger, 2018)	Turkish, Twitter & other microblogs, Manual Annotation
Taslioglu 2017	69	431	500	(Taslioglu & Karagoz, 2017)	Turkish, Twitter & other microblogs, Manual Annotation
Carvalho 2020	1.134	1.134	2.268	(Carvalho et al., 2020)	Portuguese, Farcical news, Manual Annotation
IroSvA-Spain	1.000	2.000	3.000	(Frenda & Patti, 2019)	Spanish, Twitter, Manual Annotation (?) *
IroSvA-Mexico	1.000	2.000	3.000	(Frenda & Patti, 2019)	Spanish, Twitter, Manual Annotation (?)
IroSvA-Cuba	1.000	2.000	3.000	(Frenda & Patti, 2019)	Spanish, News comments, Manual Annotation (?)
DEFT 2017 in Ghanem et al., 2020	2.425	4.882	7.307	(Ghanem et al., 2020)	French, Twitter, Manual Annotation
Ghanem 2020	6.005	5.220	11.225	(Ghanem et al., 2020)	Arabic, Twitter, Manual Annotation
Ciron	968	7.734	8.702	(Xiang et al., 2020)	Chinese, Twitter, Manual Annotation with multiple classes

ACKNOWLEDGMENT

This research was supported by the Scientific and Technological Research Council of Turkey (TUBITAK) [grant number 117E566]. The authors would like to thank Cenk Cidecio for helping us collect the data. The authors would like to thank Recep Firat Cekinel for helping us for using BERT and server infrastructure. The authors would like to thank the members of METU Data Mining Research Group for their help in data annotation.

REFERENCES

Ahmed, U., Zafar, L., Qayyum, F., & Islam, M. A. (2018, June). Irony detector at SemEval-2018 task 3: Irony detection in English tweets using word graph. In *Proceedings of the 12th international workshop on semantic evaluation* (pp. 581-586). doi:10.18653/v1/S18-1095

Alt, R., & Reinhold, O. (2020). Social CRM: Tools and Functionalities. In *Social Customer Relationship Management. Management for Professionals*. Springer. doi:10.1007/978-3-030-23343-3_3

Baloglu, U. B., Alatas, B., & Bingol, H. (2019, November). Assessment of Supervised Learning Algorithms for Irony Detection in Online Social Media. In *2019 1st International Informatics and Software Engineering Conference (UBMYK)* (pp. 1-5). IEEE. doi:10.1109/UBMYK48245.2019.8965580

Barbieri, F., Saggion, H., & Ronzano, F. (2014, June). 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). doi:10.3115/v1/W14-2609

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). *Enriching word vectors with subword information*. https://arxiv.org/abs/1607.04606

Bruntsch, R., & Ruch, W. (2017). Studying Irony Detection Beyond Ironic Criticism: Let's Include Ironic Praise. *Frontiers in Psychology*, *8*, 606. doi:10.3389/fpsyg.2017.00606 PMID:28484409

Buschmeier, K., Cimiano, P., & Klinger, R. (2014, June). An impact analysis of features in a classification approach to irony detection in product reviews. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis* (pp. 42-49). doi:10.3115/v1/W14-2608

Cambria, E., Li, Y., Xing, F. Z., Poria, S., & Kwok, K. (2020). Senticnet 6: Ensemble application of symbolic and subsymbolic AI for sentiment analysis. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 105–114. doi:10.1145/3340531.3412003

Carvalho, P., Martins, B., Rosa, H., Amir, S., Baptista, J., & Silva, M. J. (2020, March). Situational Irony in Farcical News Headlines. In *International Conference on Computational Processing of the Portuguese Language* (pp. 65-75). Springer. doi:10.1007/978-3-030-41505-1_7

Cemek, Y., Cidecio C., Ozturk A. U., Cekinel, R. F., & Karagoz P. (2020). Turkce resmi olmayan metinlerde ironi tespiti icin sinirsel yontemlerin incelenmesi [Investigating the neural models for irony detection on Turkish informal texts]. *IEEE Sinyal Isleme ve Iletisim Uygulamalari Kurultayi (SIU2020).*

Chakraborty, K., Bhattacharyya, S., & Bag, R. (2020). A Survey of Sentiment Analysis from Social Media Data. *IEEE Transactions on Computational Social Systems*, *7*(2), 450–464. doi:10.1109/TCSS.2019.2956957

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

Dülger, O. Türkçe metinlerde ironi tespiti [Irony classification in Turkish texts]. (2018). *Ulusal Yazılım Mühendisliği Sempozyumu (UYMS2018).*

Frenda, S., & Patti, V. (2019). Computational Models for Irony Detection in Three Spanish Variants. In IberLEF@ SEPLN (pp. 297-309). Academic Press.

Ghanem, B., Karoui, J., Benamara, F., Rosso, P., & Moriceau, V. (2020, April). Irony Detection in a Multilingual Context. In *European Conference on Information Retrieval* (pp. 141-149). Springer.

Irony. (2020). In *lexico.com dictionary*. Retrieved from https://www.lexico.com/en/definition/irony

Olson, R. S., Bartley, N., Urbanowicz, R. J., & Moore, J. H. (2016). Evaluation of a tree-based pipeline optimization tool for automating data science. *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference - GECCO '16*, 485–492. doi:10.1145/2908812.2908918

Pamungkas, E. W., & Patti, V. (2018, June). #NonDicevoSulSerio at SemEval-2018 Task 3: Exploiting Emojis and Affective Content for Irony Detection in English Tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation* (pp. 649-654). doi:10.18653/v1/S18-1106

Rajapakse, T. (n.d.). *PyTorch-Transformers-Classification.* GitHub repository. https://github.com/ThilinaRajapakse/pytorch-transformers-classification

Ravi, K., Ravi, V., & Prasad, P. S. R. K. (2017). Fuzzy formal concept analysis based opinion mining for CRM in financial services. *Applied Soft Computing*, *60*, 786–807. doi:10.1016/j.asoc.2017.05.028

Roeder, M. (2015, October 11). *Precision, Recall and F1 measure.* GitHub Wiki Page. https://github.com/dice-group/gerbil/wiki/Precision,-Recall-and-F1-measure

Taslioglu, H., & Karagoz, P. (2017, April). Irony detection on microposts with limited set of features. In *Proceedings of the Symposium on Applied Computing* (pp. 1076-1081). doi:10.1145/3019612.3019818

Tutaysalgir, E., Karagoz, P., & Toroslu, I. H. (2019). Clustering based Personality Prediction on Turkish Tweets. *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 825-828. doi:10.1145/3341161.3343513

Van Hee, C., Lefever, E., & Hoste, V. (2016, May). Exploring the realization of irony in Twitter data. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)* (pp. 1794-1799). Academic Press.

Van Hee, C., Lefever, E., & Hoste, V. (2016, December). Monday mornings are my fave:)# not exploring the automatic recognition of irony in English Tweets. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers* (pp. 2730-2739). Academic Press.

Vural, A. G., Cambazoglu, B. B., Senkul, P., & Tokgoz, Z. O. (2013). A framework for sentiment analysis in Turkish: Application to polarity detection of movie reviews in Turkish. In *Computer and Information Sciences III* (pp. 437–445). Springer. doi:10.1007/978-1-4471-4594-3_45

Wu, C., Wu, F., Wu, S., Liu, J., Yuan, Z., & Huang, Y. (2018, June). 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). doi:10.18653/v1/S18-1006

Xiang, R., Gao, X., Long, Y., Li, A., Chersoni, E., Lu, Q., & Huang, C. R. (2020, May). Ciron: a New Benchmark Dataset for Chinese Irony Detection. In *Proceedings of the 12th Language Resources and Evaluation Conference* (pp. 5714-5720). Academic Press.

Xu, H., Santus, E., Laszlo, A., & Huang, C. R. (2015, June). LLT-PolyU: Identifying sentiment intensity in ironic tweets. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* (pp. 673-678). doi:10.18653/v1/S15-2113

Yadav, A., & Vishwakarma, D. K. (2020). Sentiment analysis using deep learning architectures: A review. *Artificial Intelligence Review*, *53*(6), 4335–4385. doi:10.1007/s10462-019-09794-5

Zhang, S., Zhang, X., Chan, J., & Rosso, P. (2019). Irony detection via sentiment-based transfer learning. *Information Processing & Management*, *56*(5), 1633–1644. doi:10.1016/j.ipm.2019.04.006

Asli Umay Ozturk is a graduate student in both Computer Engineering and Cognitive Science departments at Middle East Technical University. They have a bachelor's degree in computer engineering and have been working on research projects focused on machine learning problems. Their works include the exploration of NLP models for several Turkish information extraction tasks and their explainability.

Yesim Cemek is a Computer Engineer who works as a Game Developer at a mobile game company. Their interests are Computer Vision, Deep Learning, NLP and Game Technologies. They studied on research projects that include Computer Vision, Deep Learning and NLP, contributed to 2 papers about Irony Detection on Turkish Informal Texts.

Pinar Karagoz is a full professor in Middle East Technical University (METU) Computer Engineering Department. She received her Ph.D. from the same department. During her doctoral studies, she worked as a researcher at SUNY Stony Brook University in New York, USA. She had research visits in MIT CSAIL, Ostrava University, Aalto University and Southern Denmark University. Her research focuses on data mining, machine learning algorithms, information retrieval, social media analysis and mining. She has publications in internationally recognized and indexed international journals including IEEE TKDE, ACM TWEB, IEEE TII and The Computer Journal, and she authored about 100 papers in international conferences in the area. In 2016 she received the best paper award in IEEE Transactions on Industrial Informatics. In 2017, her paper was nominated for Wilkes Award of the Computer Journal.