# MarSan at SemEval-2022 Task 6: iSarcasm Detection via T5 and Sequence Learners

**Maryam Najafi**[1,*], **Ehsan Tavan**[1,*]
[1] NLP Department, Part AI Research Center, Tehran, Iran
{maryam.najafi, ehsan.tavan}@partdp.ai

## Abstract

The paper describes SemEval-2022's shared task "Intended Sarcasm Detection in English and Arabic." This task includes English and Arabic tweets with sarcasm and non-sarcasm samples and irony speech labels. The first two subtasks predict whether a text is sarcastic and the ironic category the sarcasm sample belongs to. The third one is to find the sarcastic sample from a sarcastic sample and its non-sarcastic paraphrase. Deep neural networks have recently achieved highly competitive performance in many tasks. Combining deep learning with language models has also resulted in acceptable accuracy. Inspired by this, we propose a novel deep learning model on top of language models. On top of T5, the architecture uses an encoder module of the transformer, followed by LSTM and attention to utilizing past and future information, concentrating on informative tokens. Due to the success of the proposed model, we used the same architecture with a few modifications to the output layer in all three subtasks.

## 1 Introduction

Sarcasm is a sophisticated form of expression that implicitly conveys the content of a sentence. Automated sarcasm detection focuses mainly on the lexical, syntactic, and semantic levels of text analysis Hazarika et al. (2018).

Natural language understanding(NLU), dialogue systems, and text mining can benefit from sarcasm detection. Arguably, the most challenging part of sarcasm is its rarity, infrequency, difficulty in detecting, and ambiguity in meaning. Sarcasm, for instance, can imply a negative meaning with the use of positive words. For example, "Taxes are just the best, and I cannot wait to pay more 🥰 🥺 😇" a sarcastic sentence that uses positive words but carries a negative meaning of "I dislike paying taxes." As a result, detecting sarcasm poses a challenging task due to the nature of sarcastic texts, which are influenced by several factors, such as context, region, and mentality.

A sarcasm detection algorithm goes beyond sentiment analysis, and instead of looking at sentiment in a sample, it focuses on sarcasm. The purpose of this field is to identify whether a given text is sarcastic or not.

Recently, it has been shown that neural language models trained on unstructured text can implicitly store and retrieve knowledge. Studies have revealed that the Text-To-Text Transfer Transformer (T5) architecture Raffel et al. (2019) can achieve high performance for various NLP applications. An essential step in creating NLP models is choosing an appropriate embedding vector. In this research, the T5 and Multilingual T5 (MT5) (Xue et al., 2020) Encoder module for English and Arabic was implemented, respectively. Our task is comprised of three Subtasks:

- Subtask A: Predict the sarcastic nature (sarcastic or non-sarcastic) of a sample.
- Subtask B: Determine which one of the irony speech categories the sample belongs to.
- Subtask C: Given two samples, determine which one is sarcastic.

There are insufficient instances in the dataset for this task, particularly the low number of sarcastic instances which make deep learning models unsuitable for extracting text features. Fine-tunning the language model on two large open-source datasets in English and Arabic is the key to solving this challenge. A fine-tuning step may provide the model with valuable awareness of task context. For the English task, the dataset of 4,484 tweets named iSarcasm (Oprea and Magdy, 2019) was selected, while for the Arabic task, the dataset of 10,547 tweets Farha and Magdy (2020) was chosen.

[*]Equal contribution. Listing order is random.



*Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, pages 978 - 986

This research uses the t5 language model as a word embedding layer to design the sarcasm detection model. After T5, the encoder module of the transformer, the Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) layer, and scaled dot-product attention (Vaswani et al., 2017) were implemented sequentially for extracting the informative knowledge. Lastly, the model was fed an output from max-pooling. The same model has implemented all three subtasks, changing the input and output layer types.

To participate in Task 6 of SemEval-2022 Abu Farha et al. (2022), we submitted the results of 5 subtasks. The ranking of our model in subtask A ranked 7th in English and 31st[1] in Arabic. Our model placed 9th out of 22 teams in subtask B. In subtask C, our model placed 7th out of 16 teams in English and 3rd out of 13 teams in Arabic. Our code is available at GitHub[2] for researchers.

The remaining of this paper is organized as follows: Section 2 reviews related work. Section 3 describes both tasks and provided dataset. Section 4 presents the theoretical background of the proposed neural model. Implementation details are provided in Section 5, while experiments and results are presented in Section 6. Section 7 presents both quantitative and qualitative error analysis. Section 8 contains paper conclusions.

## 2 Background

It is pointed out in Javdan et al. (2020) that sarcasm can alter the meaning of a phrase, making opinion analysis error-prone. Subsequently, a model by BERT and aspect-based sentiment analysis is employed to address the issue. Based on the context dialogue sequence, this system can determine whether a response is sarcastic or not, with an F1-score of 0.73 on Twitter.

In Dadu and Pant (2020), as in Javdan et al. (2020), the researcher used two Reddit and Twitter datasets and applied Roberta-large to detect sarcasm in both datasets. González-Ibánez et al. (2011) investigates lexical (like uni-grams and dictionary-based) and pragmatic (like positive or negative emotions) features and compares the performance of machine learning techniques and human judges.

Since the meaning of sarcasm differs for each individual and may lead to misunderstandings in everyday communications, Hazarika et al. (2018) claims that user embedding can encode the stylistic and personality attributes of users, and combined with Convolutional Neural Networks (CNNs) (LeCun et al., 1999) that extracts localized information, the results are reasonable.

Kumar et al. (2020) introduce a binary classification deep learning model for sarcasm detection. Kumar et al. use Bi-LSTM and Multi-Head Attention Mechanism to obtain sentence embedding and classify input text with a softmax layer.

RoBERTa network architecture is utilized in Potamias et al. (2020) to map words onto a rich embedding space efficiently. To improve RoBERTa performance and capture temporal reliance information, use the RCNN network.

## 3 Task Description

Task 6 of SemEval-2022 presents a Sarcasm Detection dataset in English and Arabic.

Statistical information on the number of samples in the train, dev, and test data is shown in Table 1. Since there was no official dev set at the evaluation phase, we randomly selected 10% of the dataset as the dev data.

| Dataset | train | dev | test |
|---|---|---|---|
| Subtask A (English) | 3121 | 347 | 1400 |
| Subtask A (Arabic) | 2792 | 310 | 1400 |
| Subtask B | 780 | 87 | 1400 |
| Subtask C (English) | 780 | 87 | 200 |
| Subtask C (Arabic) | 670 | 76 | 200 |

Table 1: Statistical information of datasets

A description of the subtasks is provided in the following sections.

### 3.1 Subtask A

In this subtask, the model should determine the sarcastic or non-sarcastic nature of the text. Arabic and English texts can be submitted for this subtask.

Figure 1 shows the distribution of sarcastic and non-sarcastic classes in subtask A. There is an imbalance with this data, as only 25% of the samples are categorized as sarcasm, making the training process more challenging.

[1]There was an error in submitting this subtask. In the results section, we provide the true results of the proposed model for subtask A .

[2]https://github.com/MarSanTeam/Sarcasm_Detection

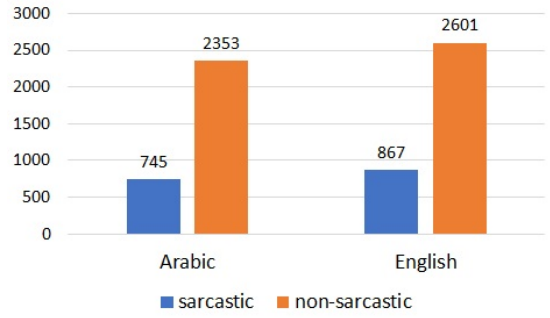



Figure 1: Label distribution in Subtask A

### 3.2 Subtask B

This subtask is a multi-label binary classification task.

Ideally, the model should predict which category of ironic speech (sarcasm, irony, satire, understatement, overstatement, and rhetorical question) input data falls into. In this subtask, all data is available in English only.

Figure 2 illustrates how irony speech categories (sarcasm, satire, understatement, overstatement, and rhetorical questions) have been distributed. A high percentage of samples are labeled sarcasm, and the lowest, with 10 samples, is for understatement.

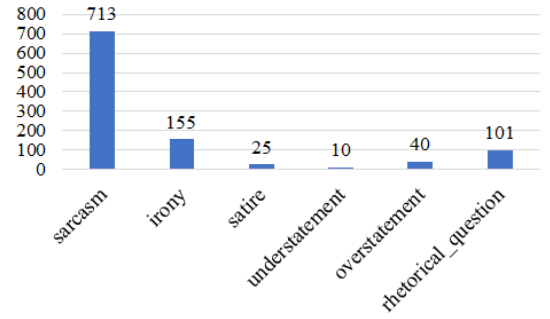



Figure 2: Label distribution in Subtask B

### 3.3 Subtask C

As part of subtask C, a sarcastic text and its non-sarcastic paraphrase is given to the model. The proposed model should determine which one is sarcastic.

In this Subtask, we have 867 samples in English and 745 samples in Arabic.

## 4 System overview

Contextual features can be extracted very efficiently with pre-trained language models. In NLP tasks, T5 has proven to be an efficient encoder-decoder framework. Using the encoder layers within the T5 language model, we can fine-tune pre-trained encoder-decoder T5 models efficiently for classification and regression tasks. Pre-trained models ease fine-tuning downstream tasks by reducing the reliance on large task-specific training datasets. Their results can be further enhanced if other deep learning architectures, such as LSTM, CNN, and attention, are applied on top of them (Tavan et al., 2021).

The framework we developed uses the encoder module of T5 and Transformer, Bi-LSTM layer, and scaled dot-product attention to determine whether a text is sarcastic. Also, we apply the same architecture to subtasks B and C but make some changes to the output and input layers, respectively. The proposed model architecture is shown in Figure3.

As part of subtasks A and B, the model gets a sequence of $S = \{s_1, s_2, s_3, ..., s_N\}$, where $s_n$ is the $n$th token of input text. In subtask C, the inputs are sequence of $P = \{p_1, p_2, p_3, ..., p_I\}$, and $Q = \{q_1, q_2, q_3, ..., q_J\}$, where $p_j$ is the $j$th token of the first text and $q_I$ is the $i$th token of the second. The input sequence $S = \{P, < /s >, Q\}$ are the final inputs for subtask C.

Subtask A estimates a sarcastic label based on the probability distribution of $P_r(y|S)$. Within subtask B, the model estimates the probability distribution $P_r(y|S)$ for each category of ironic speech. Finally, subtask C estimates the probability distribution $P_r(y|P, Q)$, predicting whether P indicates sarcasm or Q.

### 4.1 Word Representation

To obtain vector representations of input tokens, the T5 encoder is used. As mentioned, this model can obtain a contextualized embedding vector for each token in the input text by fine-tuning the T5 encoder.

### 4.2 Encoder Architecture

The encoder module of the transformer and a Bi-LSTM layer are used sequentially on top of the T5 encoder to extract the contextualized feature.

As a way to extract the relation between tokens, the encoder architecture used in the transformer Vaswani et al. (2017) employs a multi-head attention mechanism. This capability of multi-head attention allows the model to extract relationships

between input tokens, which can help identify sarcasm context.

Sarcasm detection can be enhanced by identifying contradictions and long-term dependencies in a sentence. This information can not be extracted properly from the transformer encoder because of its structure. The challenge can be tackled by using the RNNs layer to extract temporal information and long-term dependencies. Using a Bi-LSTM layer, this can be done in long sequences. The output vector of the encoder layer is calculated by concatenating the Bi-LSTM layer and the transformer encoder output.

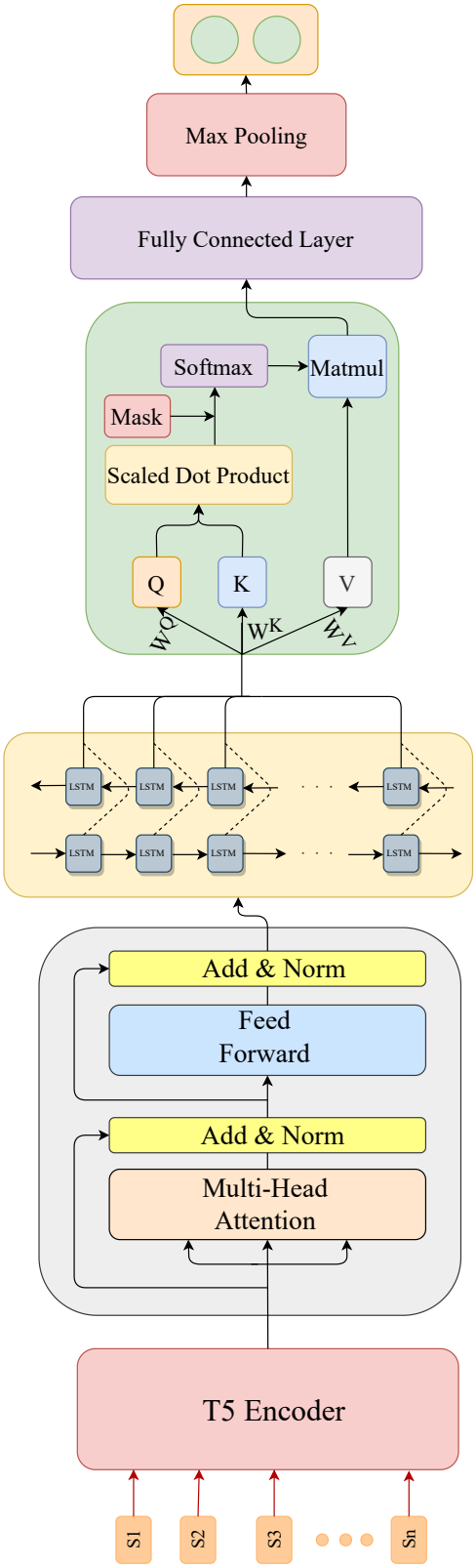



Figure 3: proposed model

### 4.3 Attention Module

In Bi-LSTM networks, close words are more likely to be correlated with the extracted attribute than words located farther away. In order to extract rich features, scaled dot-product attention assigns different weights to each token. Each token is given a weight based on its importance and relation to a class. Hence, attention can determine the relevance and importance of tokens to identify the label correctly. The scaled dot-product attention above the encoder layer enables the model to capture the importance and relationship between tokens regardless of their distance. The attention module consists of the following components:

$$W_i^Q, W_i^K, W_i^V \in R^{d_{model} \times d_k}$$
$$Q = XW_Q, K = XW_K, V = XW_V \quad (1)$$
$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Among which $W_Q$, $W_K$, and $W_V$ are trainable parameters. The input vector $X$ is multiplied by the matrices $Q$, $K$, and $V$ in order to create three matrices $Q$, $K$, and $V$. To prevent the dot-product between $Q$ and $K$ from getting too large, the dot-product between $Q$ and $K$ is divided by $\sqrt{d_k}$.

### 4.4 Prediction Module

To accurately predict sarcasm in a text by utilizing the most relevant and informative extracted features, a fully connected layer with a tanh activity function was first employed. The general representation is then obtained using a max-pooling layer from the same dimensions of different tokens. The max-pooling modules formulate in Equation 2.

$$Z = Max([h_1, ..., h_l]) \quad (2)$$

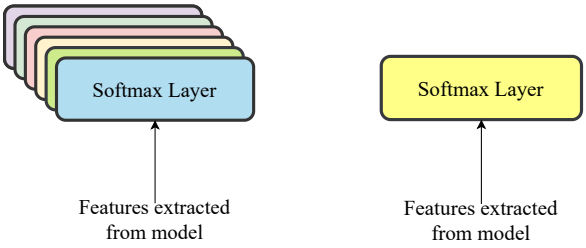
Finally, to determine the probabilities of the labels, the softmax classification method is used. The module is a simple softmax classifier that generates probabilities of distributions based on input

features. A softmax classifier is used to predict a label $\hat{y}$ from a set of discrete classes(sarcastic or not-sarcastic) for an input sequence S. The softmax classifier takes R as input:

$$P(y \mid Z) = softmax(WR + b) \quad (3)$$

$$\widehat{y} = argmax P(y \mid Z) \quad (4)$$

We used six softmax layers to predict the label of each input text in subtask B. The difference between the implementation of the proposed model in single label and multi-label is shown in Figure 4.




(a) Multi Label (b) Single Label

Figure 4: Outline of utput layer used in subtasks.

## 5 Experimental Setup

In this section, we first describe data pre-processing. Following that, we discuss the implementation details.

**Pre-processing** Data analysis revealed some samples containing URLs and user mentions. Due to the possibility of these items confusing the model in identifying the correct label, these items were removed in the pre-processing phase. Tokens such as punctuation and emojis were not removed since they could be excellent indicators of sarcasm.

**Implementation Details** PyTorch was used to implement the model, and we trained it on Nvidia V100 GPUs. Each subtask's hyper-parameters were tuned using the development set. The AdamW optimizer with the learning rate of 2e -5 is used to train the network using the back-propagation algorithm. A training method of early stopping with the patience of 5 and monitoring validation loss of sarcastic class in min mode is used. As for regularization, we evaluate the effect of the dropout on the transformer encoder and the Bi-LSTM layer; the model has a better performance when the dropout rate is set as 0.2, 0.3 respectively.

Each subtask has a batch size of 32. The number of attention heads in the transformer encoder is eight, and the hidden size in the Position-wise feed-forward Layer is 2048. One layer of Bi-LSTM with 128 LSTM units was used. Over the first fully connected layer, we applied tanh, and the output size of this layer was 256. In the case of data imbalance, the cross-entropy loss can be used with class weights. The maximum length used in T5 and MT5 tokenizers is 100. Other parameters are randomly initialized.

## 6 Results

This section reviews the different baselines and compares them with the proposed model. Due to the success of language models in recent years, we have evaluated different language models to select the most appropriate language model.

Since there was no official test data at the evaluation phase, experiments were conducted on the dev set to select the best architecture. Finally, the performance of the models on the test data is also evaluated.

### 6.1 Subtask A

Subtask A was evaluated on the F1-score of the sarcastic class. Table 2 Shows the results for subtask A. As can be seen, BERT, T5, and RoBERTa have been evaluated to determine the most appropriate language model. The T5-large and MT5-large achieve a score of 49.47% and 71.53%, respectively, outperforming other language models. It could be caused by the differences in objective learning among the models. In addition, it is worth noting that the BERT-based models predict masked words from the vocabulary and are auto-encoding models, while the T5 uses a text-to-text framework for training, and it is an auto-regressive model.

Table 3 shows the results of implementing the deep learning model. Since T5-large and MT5-large performed well in the initial experiments, they were subsequently used in subtask A. Adding transformer encoders help increase the F1-score on the sarcastic class in subtask A by capturing and focusing the most informative data on top of the language model. In English and Arabic, there was an increase in the F1-score using the transformer encoder due to the presence of multi-head attention in the transformer and the ability of this module to extract the dependencies.

Combining a Bi-LSTM layer results in a higher F1-score. This increase is due to the ability of the recurrent network to extract temporal information

| Model | Dev | | Test | |
|---|---|---|---|---|
| | sarcastic | non-sarcastic | sarcastic | non-sarcastic |
| **Subtask A (English)** | | | | |
| **T5-base** | 42.29 | 82.45 | 25.33 | 76.29 |
| **T5-large** | **49.47** | 81.05 | **32.85** | 75.89 |
| **BERT-base** | 45.89 | 76.55 | 31.40 | 73.40 |
| **RoBERTa-base** | 26.41 | 83.46 | 9.34 | 86.99 |
| **RoBERTa-large** | 39.64 | 85.92 | 25.12 | 88.59 |
| **Subtask A (Arabic)** | | | | |
| **MT5-base** | 67.88 | 89.64 | 29.12 | 56.85 |
| **MT5-large** | **71.53** | 87.23 | **31.89** | 65.33 |
| **MBERT-base** | 58.89 | 76.55 | 24.40 | 73.40 |
| **XLM-RoBERTa-base** | 58.70 | 89.80 | 29.16 | 63.32 |
| **XLM-RoBERTa-large** | 63.93 | 86.52 | 27.87 | 42.67 |

Table 2: subtask A: Evaluation result of Language models

| Model | Dev | | Test | |
|---|---|---|---|---|
| | sarcastic | non-sarcastic | sarcastic | non-sarcastic |
| **Subtask A (English)** | | | | |
| **T5-large + Transformer** | 52.41 | 81.36 | 33.68 | 75.26 |
| **T5-large + Transformer + Bi-LSTM** | 52.55 | 79.49 | 34.85 | 77.89 |
| **T5-large + Transformer + Bi-LSTM + Attention** | 55.11 | 84.48 | 40.31 | 84.53 |
| **T5-large + Transformer + Bi-LSTM + Attention + fc (ours)** | 57.18 | 87.43 | 42.51 | 83.76 |
| **ours + finetune T5** | **58.89** | 88.39 | **43.42(7)** | 84.31 |
| **Subtask A (Arabic)** | | | | |
| **MT5-large + Transformer** | 72.16 | 88.94 | 33.42 | 55.12 |
| **MT5-large + Transformer + Bi-LSTM** | 73.98 | 88.14 | 33.64 | 56.28 |
| **MT5-large + Transformer + Bi-LSTM + Attention** | 75.16 | 86.98 | **34.06** | 55.47 |
| **MT5-large + Transformer + Bi-LSTM + Attention + fc (ours)** | 75.87 | 90.34 | 31.88 | 53. 76 |
| **ours + finetune MT5 (Error in submission)** | **76.29** | 91.76 | 18.79(31) | 34.38 |
| **ours + finetune MT5 (True result)** | **76.29** | 91.76 | 32.24 | 54. 43 |

Table 3: subtask A: Baseline results. Our rank is shown in parentheses.

and long-term dependencies. Next, adding scaled dot-product attention improved the F1-score in English and Arabic, reaching 55.11% and 75.16% on the sarcastic class, respectively. A scaled dot-product attention module could improve the model's accuracy by detecting dependencies between words, which is mainly helpful in assisting the Bi-LSTM architecture in identifying spatially spaced apart words.

Finally, as mentioned before, we have also finetuned T5 and MT5 on two large datasets in English and Arabic, owing to the deficient number of samples. Using this method, the F1-scores obtained in English and Arabic have reached 58.89% and 76.29%. An improvement in the F1-score can be achieved by the task awareness of the language model.

### 6.2 Subtask B

The results of different models for subtask B are shown in Table 4. The evaluation metric in this subtask is macro F1-score. As a result of the experiments, it was found that RoBERTa achieves a macro F1-score of 11.34% on test data, which is the highest F1-score. Due to the nature of multi-label binary classification, each of the six classes in this subtask is classified by a separate classifier within the same architecture as subtask A. Our T5-based implemented model in the competition achieved the rank of 9 and F1-score of 7.43%.

| Model | Dev | Test |
|---|---|---|
| **T5-base** | 12.41 | 2.41 |
| **T5-large** | 19.68 | 5.34 |
| **Bert-base** | 18.07 | 5.76 |
| **RoBERTa-base** | 22.01 | **11.34** |
| **RoBERTa-large** | 21.28 | 9.94 |
| **ours in competition** | **24.67** | 7.43(9) |

Table 4: Subtask B: Evaluation result of Language models. Our rank is shown in parentheses.

### 6.3 Subtask C

Several experiments were performed to select the most appropriate language model for subtask C. Table 5 shows the results of implementing different models on English and Arabic. For this subtask, the evaluation metric is accuracy. Among the tested language models, on dev, the T5 and MT5 reach the highest accuracy among other language models.

For this reason, these models have been used in subsequent experiments. The results of language models are shown in Table 5.

| Model | Dev | Test |
|---|---|---|
| **Subtask C (English)** | | |
| **T5-base** | **92.24** | **72.57** |
| **T5-large** | 90.95 | 69.43 |
| **Bert-base** | 77.01 | 68.18 |
| **RoBERTa-base** | 82.75 | 68.86 |
| **RoBERTa-large** | 49.42 | 47.27 |
| **Subtask C (Arabic)** | | |
| **MT5-base** | 53.33 | 52.75 |
| **MT5-large** | **88.19** | **78.64** |
| **MBert-base** | 80.64 | 69.83 |
| **XLM-RoBERTa-base** | 45.12 | 49.73 |
| **XLM-RoBERTa-large** | 45.33 | 50.16 |

Table 5: Subtask C: Evaluation result of Language models

In Table 6, the model that was developed in this research is implemented, and the results are shown. To compete in English, the T5-large language model and its combination with the proposed deep network architecture have been used and we have reached an accuracy of 76.50%, 87.50% in English and Arabic, respectively.

| Model | Dev | Test |
|---|---|---|
| **Subtask C (English)** | | |
| **T5-base + Transformer** | 93.10 | 73.86 |
| **T5-base + Transformer + Bi-LSTM** | 94.25 | 76.64 |
| **T5-base + Transformer + Bi-LSTM + Attention** | 95.34 | 77.13 |
| **T5-base + Transformer + Bi-LSTM + Attention + fc (ours)** | 92.55 | **79.32** |
| **T5-Larg + Transformer + Bi-LSTM + Attention + fc (ours in competition)** | **93.10** | 76.50(9) |
| **Subtask C (Arabic)** | | |
| **MT5-large + Transformer** | 88.99 | 81.38 |
| **MT5-large + Transformer + Bi-LSTM** | 91.25 | 83.18 |
| **MT5-large + Transformer + Bi-LSTM + Attention** | 92.34 | 86.32 |
| **MT5-large + Transformer + Bi-LSTM + Attention + fc (ours in competition)** | **96.55** | **87.50(3)** |

Table 6: Subtask C: Proposed model result. Our rank is shown in parentheses.

## 7 Error And Performance Analysis

In this section, we analyze the performance of several components of our system. This section will examine our model's ability to correctly label samples in sarcastic or non-sarcastic contexts.

### 7.1 Subtask A

There are 1400 samples in the English test set. The model correctly identified 924 samples as non-sarcasm, which means True Negative (TN), and correctly identified 132 samples as sarcasm, known as True Positive(TP). Despite this, 68 sarcasm samples were incorrectly predicted as non-sarcasm, resulting in False Negatives(FN). Lastly, the False Positive(FP) value is very high. There were 276 input samples in this case that were incorrectly predicted as sarcasm samples. According to the results, since about 75% of the data is the negative sample, Data imbalance leads to an FP error rate of about 276 among all predictions.

**Error analysis** Some samples of post-evaluation on the model's output were examined, in Table 7. According to studies, the most significant effect on the accurate prediction of sarcasm samples is obtained from sentiment and emoji in samples (Type A, B). According to the implemented architecture, the model is more robust for long samples. However, in very short samples (Type C), weaknesses in the estimation are observed.

In addition to analyzing the strengths of the model and the factors that affected them, this section also examined its weaknesses in predicting the sample and the factors that affected them. As mentioned, a major reason is the short length of the input sample. Another reason could be the presence of misspellings (Type D) in the data and the unfamiliarity of the model with the incorrect word. A major reason for the error in estimation can be found in the absence of sentiment and emotion in the text and the fact that the sample is completely based on "human knowledge" (Type E).

As a result, the model has weaknesses in short, humane, and emotionless examples.

### 7.2 Subtask B

Detailed model results on test data for each label are presented in Table 8. Since the dataset only included one sample of the "understatement" class, the model could not correctly identify the sample's label. The "overstatement" class contains only ten samples, and due to its scarcity and complexity, the model could not predict any of these, so the F1-score on the "understatement" and "overstatement" classes is 0.

Among the remaining 4 classes, the recall value is much higher than the precision of the model, which indicates that the model performed well at

| Sample | Prediction type |
|---|---|
| **Type A: Emoji** | |
| Love it when someone with no mask chooses to sit next to me on the bus... 😑 | TP |
| Weathers wonderful today! 🌬️🌬️🌬️ | TP |
| **Type B: Sentiment** | |
| Wow the Prime Minister is so good at the telling the truth | TP |
| You on your best behaviour is me on my worst | TP |
| It makes me feel a lot safer knowing the MET Police don't investigate crimes after they happen. | TP |
| Yeah, feeding children sweets before bedtime is an awesome way of getting lots of sleep | TP |
| I swear stupid people were put on this earth to test my anger management skills | TP |
| **Type C: Short samples** | |
| Proof of unjustified victimisation! | FP |
| Rubbish | FP |
| **Type D: Misspellings** | |
| if you listen carefully, you can hear me not carig | FP |
| **Type E: Human Speech** | |
| Masks work, that's why we don't have to wear them in pubs but do in shops! | FP |
| Boris Johnson is a great leader and all his team stick to the covid rules rules | FP |
| I was waiting at the bus stop when the driver pulled up and said you waiting for a bus? I said no mate im waiting for a plane. He drove off. | FP |

Table 7: subtask A: Result analysis

| Class | Number of samples | TP | FP | FN | TN | Precision | Recall | F1-Score |
|---|---|---|---|---|---|---|---|---|
| **sarcasm** | 180 | 102 | 748 | 78 | 472 | 12.00 | 56.67 | 19.81(19) |
| **irony** | 20 | 13 | 365 | 7 | 1015 | 3.44 | 65.00 | 6.53(8) |
| **satire** | 49 | 7 | 135 | 42 | 1216 | 4.93 | 14.29 | 7.33(5) |
| **understatement** | 1 | 0 | 10 | 1 | 1389 | 0 | 0 | 0(3) |
| **overstatement** | 10 | 0 | 203 | 10 | 1187 | 0 | 0 | 0(6) |
| **rhetorical-question** | 11 | 9 | 145 | 2 | 1244 | 5.84 | 81.82 | 10.91(3) |

Table 8: Subtask B: Error analysis. Our rank is shown in parentheses.

identifying samples that belong to each class. The precision value is very low, which indicates that a significant number of positive predictions are incorrectly categorized as positive.

### 7.3 Subtask C

Table 9 provides detailed information results for English test data. According to the three evaluation metrics, the model performed well in distinguishing sarcasm samples from non-sarcasm paraphrases.

We have decided not to explain further in this section due to the model's acceptable performance and space limitation.

| Sarcasm-id | Precision | Recall | F1-Score |
|---|---|---|---|
| **0** | 79.82 | 81.31 | 80.56 |
| **1** | 78.02 | 76.34 | 77.17 |

Table 9: Subtask C: Error analysis

## 8 Conclusion

We implement a novel deep learning approach based on language models. The last hidden state of T5 is used as an embedding layer in this architecture. On top of this layer, a bidirectional LSTM is used to extract future and past contexts as representations of the input text. LSTM output is processed using an attention mechanism, which focuses more on the valuable tokens to predict.

The main challenge in this study was that there were not enough samples in the dataset. To solve this, we fine-tuned the T5 language model with other large open-source datasets to have a language model that had a pre-awareness of the task and a higher accuracy. This fine-tuned language model was then used as an embedding representation in our deep architecture.

To evaluate the performance of the developed model and find the best language model, many experiments were conducted. The experimental results show that the T5 language model covers a reasonable range of results and is most appropriate for our architecture.

The same architecture was used in all three subtasks, as mentioned earlier. Due to the multi-label nature of subtask B, six separate classifiers were used instead of one to produce the output. We identified specific sarcasm challenges through error analysis, creating immediate future tasks.

As a final point, our architecture has already

been created for English and Arabic, but it could be easily extended to other languages.

## References


Ibrahim Abu Farha, Silviu Oprea, Steven Wilson, and Walid Magdy. 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)*. Association for Computational Linguistics.

Tanvi Dadu and Kartikey Pant. 2020. Sarcasm detection using context separators in online discourse. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 51–55.

Ibrahim Abu Farha and Walid Magdy. 2020. From arabic sentiment analysis to sarcasm detection: The arsarcasm dataset. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, pages 32–39.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 581–586.

Devamanyu Hazarika, Soujanya Poria, Sruthi Gorantla, Erik Cambria, Roger Zimmermann, and Rada Mihalcea. 2018. Cascade: Contextual sarcasm detection in online discussion forums. *arXiv preprint arXiv:1805.06413*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Soroush Javdan, Behrouz Minaei-Bidgoli, et al. 2020. Applying transformers and aspect-based sentiment analysis approaches on sarcasm detection. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 67–71.

Avinash Kumar, Vishnu Teja Narapareddy, Veerubhotla Aditya Srikanth, Aruna Malapati, and Lalita Bhanu Murthy Neti. 2020. Sarcasm detection using multi-head attention based bidirectional lstm. *Ieee Access*, 8:6388–6397.

Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. 1999. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer.

Silviu Oprea and Walid Magdy. 2019. isarcasm: A dataset of intended sarcasm. *arXiv preprint arXiv:1911.03123*.

Rolandos Alexandros Potamias, Georgios Siolas, and Andreas-Georgios Stafylopatis. 2020. A transformer-based approach to irony and sarcasm detection. *Neural Computing and Applications*, 32(23):17309–17320.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Ehsan Tavan, Ali Rahmati, Maryam Najafi, and Saeed Bibak. 2021. Bert-dre: Bert with deep recursive encoder for natural language sentence matching. *arXiv preprint arXiv:2111.02188*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.