ARC-NLP at CheckThat! 2022: Contradiction for Harmful Tweet Detection

Cagri Toraman¹, Oguzhan Ozcelik¹, Furkan Şahinuç¹ and Umitcan Sahin¹

¹Aselsan Research Center, Ankara, Turkey

Abstract

The target task of our team in CLEF2022 CheckThat! Lab challenge is Task-1C, harmful tweet detection. We propose a novel approach, called ARC-NLP-contra, which is a contradiction check approach by using the idea that harmful tweets contradict with the real-life facts in the scope of COVID-19 pandemic. Besides, we propose and examine two other models. The first model, called ARC-NLP-hc, is a traditional approach that utilizes hand-crafted tweet and user features. The second model, called ARC-NLP-pretrain, pretrains a Transformer-based language model by using COVID-related Turkish tweets. We compare the performances of these three models, and submit the highest performing model in the preliminary experiments to the challenge. We make submissions for Task-1A, 1B, 1C in Turkish and Task-1C in English. We have the winning solution for Task-1C, harmful tweet detection in Turkish, using ARC-NLP-contra that is our contradiction check approach.

Keywords

Claim Detection, Contradiction, COVID-19, Harmful Tweet Detection, Language Model, Tweet, Worthiness Checking.

1. Introduction

The effects of the COVID-19 pandemic maintain its existence for more than two years. The power of the social media is experienced more intensely when such global events break out. Although people can get news and collaborate via social platforms in during such events, undesired behaviors can also take place in these platforms such as spreading misinformation, along with false claims and sharing harmful content. In this regard, CLEF2022 CheckThat! Lab organizes a challenge that focuses on the detection of such undesired behaviors [1, 2, 3, 4, 5]. The lab consists of three main tasks that are Task-1: Identifying Relevant Claims in Tweets, Task-2: Detecting Previously Fact Checked Claims, and Task-3: Fake News Detection. We submit our solutions to Task-1 in this challenge.

The target task of our team, called ARC-NLP (Aselsan Research Center - Natural Language Processing team), in this challenge is Task-1C, harmful tweet detection, in Turkish. We propose a novel model, called ARC-NLP-contra, which is a contradiction check approach by using the idea that harmful tweets contradict with the real-life facts in the scope of COVID-19 pandemic. In other words, we assume that harmful tweets spread misinformation. We thereby check the claims in tweets with a manually generated fact list by using reliable sources, such as

CLEF 2022: Conference and Labs of the Evaluation Forum, September 5–8, 2022, Bologna, Italy

Corraman@aselsan.com.tr (C. Toraman); ogozcelik@aselsan.com.tr (O. Ozcelik); fsahinuc@aselsan.com.tr (F. Şahinuç); ucsahin@aselsan.com.tr (U. Sahin)

^{© 2022} Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

Government Offices and UNICEF. Our model placed first in the leaderboard for Turkish. Since this approach is language-independent, i.e. one can create other fact lists in other languages, we also submit our winning solution to Task-1C in English. However, our solution does not perform as expected in English, possibly due to our machine-translated fact list from Turkish to English.

We also propose and examine two other approaches for Task-1C, harmful tweet detection. The first model, called ARC-NLP-hc, is a traditional approach that utilizes hand-crafted tweet and user features. The second model, called ARC-NLP-pretrain, pretrains a Transformer-based language model, similarly to BERTweet [6], by using COVID-related Turkish tweets that we have collected during the training stage of this challenge. We call our pretrained language model as RoBERTweetTurkCovid. To the best of our knowledge, RoBERTweetTurkCovid is the first Transformer-based language model pretrained on Turkish tweets, which can help other researchers to fine-tune models on various downstream tasks.

We compare the performances of these three models in the preliminary experiments, and submit the highest performing model, ARC-NLP-contra, to Task-1C. Besides, we make submissions to other subtasks, Task-1A (check-worthiness of tweets) and Task-1B (verifiable claim detection). Since our contradiction check approach is not directly applicable to other subtasks, we submit the results of ARC-NLP-pretrain for Task-1A and 1B in Turkish, since we train this language model by using a Turkish corpora.

Overall, we make submissions for Task-1C in both Turkish and English; and for Task-1A and 1B in Turkish. We have the winning solution for Task-1C, harmful tweet detection in Turkish, using ARC-NLP-contra that is our contradiction check approach. In this paper, we explain our proposed models; namely (i) ARC-NLP-hc with hand-crafted features, (ii) ARC-NLP-pretrain with a novel pretrained language model, and (iii) ARC-NLP-contra with a novel contradiction check approach. We then report the results of our preliminary experiments, along with our results in the final leaderboard.

2. Methods

2.1. Model-1: ARC-NLP-hc

In order to perform statistical analysis of the challenge data, we extract hand-crafted features to classify the tweets into predefined classes, such as harmful or not in Task-1C. We extract the following hand-crafted features including tweet content and user attributes:

- N-grams (bigrams and trigrams),
- Hashtags,
- User features.

These features both separately and in combination are used in classifying whether a tweet is harmful or not. In the following subsections, the extraction process is detailed for each feature.

N-grams: We use bigrams and trigrams to extract relevant features from the data. For the preprocessing step, we first transform Turkish characters (i.e., "çğıöşü") to English characters

The first 5 bigrams and trigrams of 1C Turkish harmful and unharmful data (including their translations
in English) ranked according to their frequencies.

Harmful	Translation	Freq.	Unharmful	Translation	Freq.
doz asi	vaccine dose	.0144	doz asi	vaccine dose	.0134
asi karsiti	vaccine opposer	.0096	asi karsiti	vaccine opposer	.0089
kalp krizi	heart attack	.0096	doz biontech	biontech dose	.0059
doz biontech	biontech dose	.0085	pcr testi	PCR test	.0057
pcr testi	PCR test	.0074	asi olun	get vaccinated	.0041
bilim kurulu uyesi	sci. board member	.0027	iki doz asi	two dose vacc.	.0013
acil kullanim olayi	emerg. use auth.	.0027	haftada vaka sayisi	weekly num. cases	.0009
yeni dunya duzeni	new world order	.0021	son haftada vaka	cases in last week	.0009
yanlislikla covid asisi	vacc. by mistake	.0021	iki doz sinovac	two dose sinovac	.0009
asi karsiti degiliz	not against vacc.	.0021	yerli asimiz turkovac	natl. vacc. turkovac	.0009

(i.e., "cgiosu"). We then remove all URLs from the tweets. Furthermore, we also remove non alphabetical characters such as emojis and placeholders in the text. We remove punctuation and lowercase all the tweets. For the Turkish stop words, we use the collection list given in [7], which are disregarded when forming bigrams and trigrams.

To construct n-gram table for the data, we divide bigrams and trigrams extracted from tweet text into the harmful and unharmful classes. Table 1 shows the first 5 of bigrams and trigrams ranked according to their number of occurrences in the data. Since the data is imbalanced with respect to the classes (i.e., there are more unharmful data instances than harmful ones), we also compute each n-gram's normalized frequency within their own category, which are given in Table 1. The frequencies are computed dividing an n-gram's total number of occurrences by the number of all n-gram occurrences in its corresponding category. For instance, "*doz asi*" occurs 27 times in tweets that are labeled harmful. Since there are 1881 harmful n-gram occurrences in the data, its normalized frequency is computed as 0.0144.

We use the tweets in the data to construct the n-gram table \mathcal{T}_{1}^{ngram} (\mathcal{T}_{1}^{ngram} : harmful, \mathcal{T}_{2}^{ngram} : unharmful) aforementioned in the previous step. Then, for the tweet *i* in the data, we allocate two features denoted by $f_{1,i}^{ngram}$ and $f_{2,i}^{ngram}$. We compute these features as follows:

$$f_{1,i}^{ngram} = \sum_{j} n_{i,j} w_{1,j}, \ w_{1,j} \in \mathcal{T}_{1}^{ngram},$$
$$f_{2,i}^{ngram} = \sum_{j} n_{i,j} w_{2,j}, \ w_{2,j} \in \mathcal{T}_{2}^{ngram},$$

where j is an extracted n-gram from the tweet i. We extract bigrams and trigrams in this study. $w_{1,j}$ and $w_{2,j}$ are the corresponding normalized frequency values in n-grams tables \mathcal{T}_1^{ngram} and \mathcal{T}_2^{ngram} , respectively. $n_{i,j}$ is the number of times n-gram j occurs in tweet i. In other words, each tweet in the data is represented with two-dimensional feature vector, whose dimensions correspond to harmful and unharmful n-gram occurrences extracted from the data. As seen from Table 1, there are some contradictory n-grams in the data. For example, "*asi karsiti*" translated

The most frequently used first 8 hashtags that belong to harmful and unharmful categories (including their translations in English) in Task-1C Turkish dataset

Harmful #	Translation	Freq.	Unharmful #	Translation	Freq.
pcrdayatmasidurdurulsun	stop PCRs	.0722	biontech	-	.0511
biontech	-	.0500	turkovac	-	.0365
asivepcrdurdurulsun	stop vacc.	.0278	pcrdayatmasidurdurulsun	stop PCRs	.0292
kalpkrizlerisalgini	heart diseases	.0278	coronavirus	-	.0292
denekolmaturkiye	no tests	.0278	covid19	-	.0146
pcrhataliasizararli	harmful vacc.	.0278	sondakika	newsbreak	.0122
nepcrneasi	harmful vacc.	.0167	biontechyanetki	side effect	.0097
biontechyanetki	side effect	.0111	asihayatkurtarir	vacc. saves lifes	.0073

as "*vaccine opposer*" can be somehow thought to be a harmful bigram; however, it's normalized frequency values in both categories (i.e., .0096 vs .0089) are very close. This suggests that n-gram features might not suffice on their own to classify harmful tweets correctly.

Hashtag Analysis: Similarly to n-gram feature extraction, we perform hashtag analysis on the data. For the preprocessing step, we first transform Turkish characters (i.e., "çğiöşü") to English characters (i.e., "cgiosu"). We then lowercase all the tweets in the data. Similarly to the n-gram feature analysis, we divide the data into two categories: harmful and unharmful. Table 2 shows the most frequently used first 8 hashtags that belong to harmful and unharmful classes. There are 180 and 411 hashtags in harmful and unharmful classes, respectively. As given in Table 2, some hashtags belong to both classes; thus, their frequencies (i.e., the number of occurrences) in the data should be considered when extracting hashtag related features. We compute each hashtag's normalized frequency by dividing its total number of occurrences by the number of all hashtags in its corresponding category. For instance, hashtag "*pcrdayatmasidurdurulsun*" occurs 13 times in tweets that are labeled harmful. Since there are 180 harmful hashtags in the data, its normalized frequency is computed as 0.0722.

We extract hashtags from the data to construct the hashtag table \mathcal{T}^{tag} (\mathcal{T}_1^{tag} : harmful, \mathcal{T}_2^{tag} : unharmful). Then, similarly to n-gram normalized frequency analysis, we compute each hashtag's normalized frequency within its own category. For the tweet *i*, we allocate two features denoted by $f_{1,i}^{tag}$ and $f_{2,i}^{tag}$. We compute these features as follows:

$$f_{1,i}^{tag} = \sum_{j} w_{1,j}, \ w_{1,j} \in \mathcal{T}_1^{tag},$$
$$f_{2,i}^{tag} = \sum_{j} w_{2,j}, \ w_{2,j} \in \mathcal{T}_2^{tag},$$

where *j* is an extracted hashtag from the tweet *i*. $w_{1,j}$ and $w_{2,j}$ are the corresponding normalized frequency values in hashtag tables \mathcal{T}_1^{tag} and \mathcal{T}_2^{tag} , respectively. As opposed to n-gram feature extraction, we do not consider the number of times a hashtag occurs in a given tweet. In this way, the same hashtags that occur multiple times in a given tweet are disregarded and counted as one. The problem of contradictory features is also present in hashtag feature extraction. For

Tweet ID	Age	Influential	Verified	Status	Favorites
1423282834987372553	0.0	0.6357	0.0	0.0	0.0
1428301747110522881	0.3333	0.5802	0.0	0.0012	0.0118
1443304873471221760	0.1966	0.6464	0.0	0.0782	0.1626
1421163651084562438	0.2133	0.7029	0.0	0.0018	0.0066
1428447916826501126	0.0188	0.7449	0.0	4.49e-05	0.0003

Table 3Sample user features that we extract from Twitter API's public access.

instance, "*biontechyanetki*", translated as "*biontech side effects*", might be considered as a harmful hashtag; but, it also frequently occurs in the tweets that are labeled unharmful. Furthermore, the cold start problem is also prevalent in hashtag extraction; since only a portion of the tweet data includes hashtags and extracted hashtags from the training data might not be present in the test data and vice versa [8]. Therefore, hashtag features might not be enough to classify harmful tweets on their own.

User Features: In order to obtain user features, we collect additional data from Twitter API's public access by using tweet ids. Since some users or tweets are deleted, we cannot extract user features for all tweets. A set of sample extracted features are shown in Table 3. These user features are explained as follows.

- **Age**: This feature indicates account age, i.e., the time passed between creating the account and sending the tweet. If the age feature is not extracted from Twitter API, then the value is set to 0. The age values are normalized in all data.
- **Influential**: This feature indicates account's logarithmic ratio of followees to followers. If the influential feature is not extracted from Twitter API, then the value is set to 0. The influential values are normalized in all data.
- **Verified**: This feature indicates whether the user account is verified or not. If verified, then the value is set to 1; otherwise, 0.
- **Status**: This feature indicates account's total number of tweets, retweets, and replies. If status feature is not extracted from Twitter API, then the value is set to 0. The status values are normalized in all data.
- **Favorites**: This feature indicates account's total number of favorites. If favorite feature is not extracted from Twitter API, then the value is set to 0. The favorite values are normalized in all data.

As our classification model, we use XGBoost, an optimized distributed gradient boosting library that is highly efficient, flexible, and widely used in machine learning applications [9]. The XGBoost model that we employ in Task-1C is trained with the hand-crafted features (i.e., n-gram, hashtag, and user features) separately and in combination.

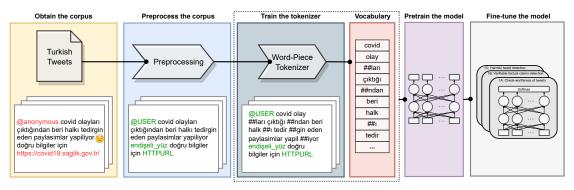


Figure 1: ARC-NLP-pretrain model diagram. We collect and preprocess COVID-related Turkish by replacing tweet-specific tags such as usernames, links, and emojis. The vocabulary is obtained by training the Word-Piece tokenizer with the preprocessed text. We then use the RoBERTa-base architecture to pretrain our language model. The pretrained model can be fine-tuned for several tasks, such as check-worthiness of tweets and harmful tweet detection. The translation of the sample tweet in the figure is "@USER Since the covid incidents, there have been posts that have disturbed the public *[worried face emoticon]* for correct information HTTPURL".

2.2. Model-2: ARC-NLP-pretrain

Considering the domain of this challenge (i.e. COVID-19 pandemic), we argue that pretraining of a Transformer-based language model, called **RoBERTweetTurkCovid**, by using COVID-related Turkish tweets can be more effective than using existing language models pretrained on variety of text documents such as BERTurk [10], or English tweets such as BERTweet [6].

RoBERTweetTurkCovid-b-30k-wp¹ is our uncased pretrained model whose corpus contains COVID-19 pandemic related Turkish tweets (*b* refers to base model, *30k* refers to the vocabulary size, and *wp* refers to the WordPiece tokenizer). Figure 1 describes the phases of obtaining our pretrained language model. We collect and preprocess COVID-related Turkish tweets by replacing tweet-specific tags such as usernames, links, and emojis. After preprocessing step, we train a WordPiece tokenizer and obtained our vocabulary contains 30k tokens. Since harmful tweet detection is a text sequence classification task, we consider a base model with optimized pretraining objective scheme, e.g., removing next sentence prediction. We therefore use the RoBERTa-base architecture [11] instead of other Transformer-based language models, such as BERT [12], to pretrain our language model. Finally, our pretrained model is fine-tuned for different downstream tasks, i.e. Task-1A, 1B, and 1C in Turkish.

Tweet Collection and Preprocessing We collect 5,851,000 Turkish tweets from Twitter API's academic access. We use Twitter API's language field to obtain Turkish tweets. In order to satisfy that tweets are related to the COVID-19 pandemic, we create a COVID-related list with 75 keywords, as given in Table 4. The keywords are selected by the authors considering commonly used COVID-related words and phrases in Turkish. Note that we have different versions of the same keyword, since Turkish users write COVID-19 in different forms such as "kovid" and "covid19". Besides, Twitter API's academic access does not allow regular expressions,

¹Our models will be published at https://huggingface.co/ctoraman

we therefore add different forms of the same phrase such as "aşı oldum" (translated as "I got vaccinated") and "aşı oldun" (translated as "you got vaccinated").

Table 4

		Keywords			
sosyal mesafenin sosyal mesafeye covid aşısından covid aşısının koronavirüsün coronavirüsün koranavirüsün sosyal mesafe #sosyalmesafe sosyalmesafe coranavirus koronavirus	koranavirus coronavirus kovid aşısı covid aşısı aşı olmanın aşı olmasak aşı olanlar pandemiden pandeminin pandemiyle aşı olsana aşı olalım aşı olmalı	aşı olunca pandemiye koronadan coronayla aşı oldum aşı oldun aşı olmak aşı olsak izolasyon maskeleri #covid-19 #kovid-19	covid-19 kovid-19 covidden kovidten covidten aşı olma maskenin maskeden evde kal #evdekal #covid19 #kovid19	<pre>#pandemi pandemi covid19 kovid19 covidle kovidle evdekal maskeyi #korona #corona korona corana</pre>	korana aşıdan aşının aşıyla #covid covid kovid aşıyı maske

The keyword list used for the collection of COVID-19 pandemic related Turkish tweets. Keywords are sorted according to their length.

After obtaining tweets, we filter out near-duplicate tweets by using the Dice similarity with 85% threshold [13]. We empirically set the threshold value by investigating the outputs of the preliminary experiments. We then preprocess them following a similar approach to BERTweet [6]. We replace user mentions with @USER, and external links with HTTPURL. We also replace emoji with their text descriptions in Turkish. For that, we use the emoji package² in Python. Since it does not support Turkish, we create an emoji list with Turkish descriptions, and modify the package to run with this list. We lastly replace the new-line character with the space character.

Pretraining The model architecture is similar to RoBERTa-base [11]. There are 12 layers and 12 attention heads with a hidden dimension size of 768. Following [14], we apply WordPiece [15] tokenizer with a vocabulary size of 30k. The pretraining details of our base model is given in Table 5. For comparison, we also provide the details of BERTurk model [10], which is a Turkish pretrained version of BERT-base [12].

We use AdamW [16] optimizer (β_1 is 0.90, β_2 is 0.98, and ϵ is 1e-6), linear scheduling with a warmup ratio of 1e-2 and peak learning rate of 5e-5, and gradient accumulation with 22 steps. Other hyperparameters are set to the RoBERTa configuration [11].

2.3. Model-3: ARC-NLP-contra

When we examine the dataset, we notice that harmful tweets display similar patterns. In general, harmful tweets consist of misinformative content about COVID-19 such as conspiracy theories

²https://github.com/carpedm20/emoji/

Configuration	BERTurk-base	RoBERTweetTurkCovid
Total parameters	110.62 M	108.79 M
Train data	35 GB	1 GB
Layers	12	12
Heads	12	12
Hidden size	768	768
Batch size	n/a	264
Max length	512 tokens	514 tokens
Train time	9.63 days	12 hours
Hardware	TPU v3-8	4x Nvidia RTX2080 Ti

Details of pre-training configurations for BERTurk and our model, *RoBERTweetTurkCovid*. (*Train time and hardware are given for a vocabulary size of 30k tokens.

about pandemic or the vaccines' side-effects that do not exist. The common ground of the harmful tweets is that they contradict with the facts about COVID-19. At this point, we convert the problem from harmful tweet detection to contradiction detection.

Detection of misinformative or harmful tweets is a difficult task due to several reasons. Although current state-of-the-art language models can capture contextual relations in a sentence [12], they might not able to distinguish the true information from the false one by only looking at the sentence itself. There is a need for a reference point (i.e. fact) to find out which information is true. For instance, the sentence "COVID-19 vaccines include HIV virus." does not give significant information about misinformation to the language models. It is also difficult to find a pattern among harmful tweets. On the other hand, providing a fact for every single instance in the dataset is not a feasible solution. We therefore propose to feed the facts to the language model in our contradiction method.

ARC-NLP-contra consists of three main stages, as illustrated in Figure 2. First, we extract important facts about the COVID-19 pandemic. Then, we associate those facts with every data instance. Finally, we train a language model based on contradiction detection.

2.3.1. Fact extraction

We gather COVID-related facts to provide more knowledge to the model. For reliability, we extract a list of facts from Turkish Ministry of Health³⁴ and Unicef Turkey⁵. These facts are published in order to protect the society from misinformation spread regarding the COVID-19 pandemic. Some examples along with their English translations are given as follows:

• Covid-19 aşısı ile insanlara mikroçipler yerleştirileceği iddiasının bilimsel bir dayanağı yok. (There is no scientific basis for the claim that microchips will be implanted in people with the Covid-19 vaccine.)

³https://covid19asi.saglik.gov.tr/TR-77694/sikca-sorulan-sorular.html

⁴https://covid19.saglik.gov.tr/TR-66125/sikca-sorulan-sorular-halka-yonelik.html

⁵https://www.unicef.org/turkey/gercek-mi-efsane-mi-koronavirus-covid-19-hakkinda-ne-kadar-bilgi-sahibisiniz

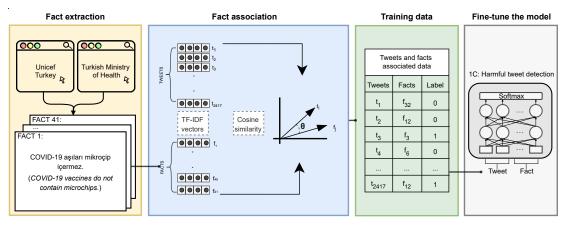


Figure 2: ARC-NLP-contra model diagram. First, general facts related to COVID-19 pandemic are gathered from reliable sources. In fact association, TF-IDF vectors are obtained for facts and tweets. For each tweet, the fact with the highest cosine similarity is determined. After all pairs are determined. A Transformer based language model is fine-tuned as contradiction detection task.

- Aşı olanlara HIV virüsü bulaşmıyor ya da aşılananlar bu enfeksiyona daha açık hale gelmiyor. (Those who are vaccinated do not become infected with HIV, or those who are vaccinated do not become more susceptible to this infection.)
- Virüs yüksek hava sıcaklığında da düşük hava sıcaklığında bulaşabilmektedir. (The virus can be transmitted at high air temperature or low air temperature.)

As observed from examples, facts include the information that mostly aims to correct the misinformation about vaccines and COVID-19. We extract 41 facts in total.

2.3.2. Fact association

Second phase of the ARC-NLP-contra method is to associate facts with related tweets. In order to associate facts and tweets, we check the similarity between two sentences (i.e. the sentences of a fact and tweet). We use the Cosine similarity measurement, and choose the fact that has the highest similarity score with a given tweet. We use two methods for encoding sentences into vector representations before calculating the Cosine similarity scores. We first utilize a deep learning approach, SBERT model for similarity [17]. Unlike original BERT model [12], SBERT model is more appropriate for the Cosine similarity for obtained sentence embeddings. The second method is to encode sentences with conventional TF-IDF vectors.

In the submitted model, we use the TF-IDF vectors. The reason of this choice is that SBERT focuses on semantic similarity, while TF-IDF focuses on syntactic similarity (i.e. the bag-of-words model). For instance, although a tweet and its related fact are in the same context (e.g., effectiveness of the vaccines), they can have opposite arguments (i.e. they are not semantically similar), which may result in less similarity. In this case, a harmful tweet may be associated with a fact of different context. However, our aim is to find the fact with the same context as harmful tweet rather than finding the most semantically similar fact. On the other hand, the bag-of-word model with TF-IDF vectors can be more immune to this opposite meaning drawback. Since it

determines the similarity according to existence of the same words, facts and harmful tweets that are in the same context can be matched without considering semantic similarity between them. Furthermore, our early experiments that include SBERT in fact association phase do not give promising results. Therefore we follow the mentioned procedure only with TF-IDF method. Note that we do not detect harmful tweets in this phase, but associate a possibly related fact with our tweets. We detect harmful ones in the next phase called contradiction detection.

While extracting TF-IDF features of the tweets for fact matching, we follow some preprocessing steps. First, we remove URLs and punctuation from the tweets. We make each word lowercased. Since TF-IDF considers word counts, these steps are useful for reducing possible noise. The same steps are also applied to fact sentences. After extracting features for all facts and tweets, the fact giving the highest cosine similarity is chosen for each tweet.

2.3.3. Contradiction detection

After providing facts implicitly, we approach to the problem as a kind of Natural Language Inference. In conventional Natural Language Inference, the task is to determine veracity of a hypothesis for a given premise. If the hypothesis is true, the relation between the premise and the hypothesis is named as entailment. If there is no relation between the premise and hypothesis, the relation becomes neutral. Lastly, the relation is considered as contradiction when hypothesis is not correct for the given premise. The sentences, "I have been in Turkey for a week." and "I visited Eiffel Tower yesterday." constitute an example for the contradiction.

In our problem, we try to create a contradiction relationship between the facts and harmful tweets, then to solve it via text classification. Since harmful tweets often contain statements that contradict general facts about COVID-19, we expect that such modelling can be more effective than standard text classification. Furthermore, we provide the reference information which is a critical aspect for the misinformation tasks with this method. At this point, it should be noted that we treat neutral and entailment categories as a single class because the main problem is a binary classification.

In classification, we utilize the state-of-the-art Transformer-based language models. We imported a pretrained model and fine-tune it via the harmful tweet dataset. We feed tweet and fact pairs as data instances separated by *[SEP]* token. We do not make any changes in labels because the pairs with the harmful tweets automatically indicate a contradiction.

3. Experiments

3.1. Datasets

The challenge task includes the datasets in six languages that are Arabic, Bulgarian, Dutch, English, Spanish and Turkish [1, 2, 3]. The datasets contain a training, development, development test splits. In the last part of the challenge, unlabeled split (Test) is released. The subtasks and data statistics are given in the following subsections. The dataset statistics are given for all tasks in Table 6.

Task	1A-TR	1B-TR	1C-TR	1C-EN
Train	2,417	2,417	2,417	3,323
Development	222	222	222	307
Development Test	660	660	660	910
Test	303	512	512	252
Positive class ratio	0.17	0.65	0.24	0.09

The number of instances and ratio of the positive instances in Task-1A, 1B, and 1C Turkish (TR) dataset and 1C English (EN) dataset.

Task-1A: Check-worthiness of tweets The aim of Task-1A is to detect whether a tweet is worthy for fact-checking. This classification task is defined with binary labels, worthy or not worthy. Model performances are measured by the F1 score of positive class for this task.

Task-1B: Verifiable factual claims detection The aim of the Task-1B is to detect a tweet whether it contains a verifiable factual claim. This classification task is defined with binary labels, claim or no claim. Model performances are measured by the accuracy metric for this task.

Task-1C: Harmful tweet detection The aim of Task-1C is to detect harmful tweets for society. Model performances are measured with the F1 score of the positive class for this task.

3.2. Experimental Setup

In this section, we explain the experimental setups for the preliminary experiments of each model.

3.2.1. Model-1: ARC-NLP-hc

When extracting n-gram and hashtag features, we construct n-gram and hashtag tables \mathcal{T}^{ngram} and \mathcal{T}^{tag} using the tweet data in the Training and Development splits. We train the classifier with the features extracted from these tables as explained previously. We then report the classification results only on the Development Test data.

For the user features, the collected tweet json objects from Twitter API are 1766 (out of 2417) for the Training split, 166 (out of 222) for the Development split, and 499 (out of 660) for the Development Test split. The reason for missing objects are that some user accounts and tweets are deleted or suspended.

The n-gram (N), hashtag (H), and user (U) features might not suffice on their own to achieve a good classification performance due to the problems of contradictory features and cold start. Therefore, we also report classification results on the combined features.

The classifier is trained with the user features collected from the Training and Development splits, while the Development Test split is spared for evaluation and reporting classification results for our preliminary experiments.

As the classifier, we use XGBoost [9] with a grid-search hyperparameter optimization. The grid-searched hyperparameters are as follows:

- *max_depth* (Maximum depth of a tree): [3, 6, 10],
- *learning_rate*: [0.001, 0.01, 0.05, 0.1],
- *n_estimators* (Number of estimators): [100, 500, 1000],
- colsample_bytree (Subsample ratio of columns when constructing each tree): [0.3, 0.7, 0.9].

The classifier optimizes each hyperparameter for the hand-crafted features separately and in combination. The optimized hyperparameter list for each feature is given in Table 7.

Table 7

Hyperparameter list for our model trained with hand-crafted features: (N)-grams, (H)ashtags, (U)sers, separately and in combination

Model	max_depth	learning_rate	n_estimators	colsample_bytree
ARC-NLP-hc-N	6	0.1	100	0.3
ARC-NLP-hc-H	3	0.05	100	0.3
ARC-NLP-hc-U	3	0.01	100	0.3
ARC-NLP-hc-NH	6	0.1	100	0.3
ARC-NLP-hc-NU	3	0.05	100	0.7
ARC-NLP-hc-HU	3	0.0	100	0.7
ARC-NLP-hc-NHU	6	0.1	500	0.3

3.2.2. Model-2: ARC-NLP-pretrain

During the fine-tuning of our Transformer-based language model, **RoBERTweetTurkCovid**, we employ PyTorch 1.11 [18]. In our preliminary experiments of this model, we use the Training split without making any changes on the given dataset, and evaluate the model on the Development Test split. We use the predefined hyperparameters for the fine-tuning process of our model. We use a batch size of 8 instances, and run training for 5 epochs with a learning rate of 5e-5.

3.2.3. Model-3: ARC-NLP-contra

While extracting TF-IDF features for ARC-NLP-contra, we do not put any constraint on the number of features. Therefore, the number of TF-IDF features correspond to the number of unique words in the dataset. Since the dataset is quite small, the number of features does not create any computational overhead. Shrinking the already small features is also not a reasonable approach for the sake of information loss.

In fine-tuning the language model for ARC-NLP-contra, we use the BERTurk model [10], which is a BERT-base model pretrained on Turkish corpora. We utilize Trainer API from Huggingface library [19]. We conduct a hyperparameter search by utilizing Optuna along with Trainer [20]. We optimize batch size, number of epochs, and learning rate. In total, we search hyperparameters along 20 trials. The number of epochs is searched in the interval [5, 15]. Batch

size is searched in the set {4, 8, 16, 32}. Learning rate is searched in the interval [1e-5,1e-4]. The best hyperparameters are chosen by F1 score of positive class on the Development split. The best hyperparameters are 8 for batch size, 15 for epochs and \approx 2.5e-5 for learning rate. For our submission to English, we use translations of the Turkish facts to English by Google Translate.

3.3. Experimental Results

In this section, we report our preliminary results obtained from the labeled Development Test split. We choose which model to submit by using the results of our preliminary experiments. We also report our best model's leaderboard results. In addition, we present results of baseline methods determined by the organizers.

3.3.1. Baseline Results

In this part, we report the evaluation results of the baseline methods for Task-1 determined by the organizers⁶. These methods are as follows: 1) random prediction, 2) majority, and 3) TF-IDF. 10 random seed initializations are performed for the random baseline. The best single results along with the average of 10 seeds are given in Table 8.

Table 8

Baseline results for the tasks Turkish 1A-1B-1C and English 1C. Results are presented in terms of F1 score of positive class for the Tasks 1A and 1C, and accuracy for the Task-1B.

	Task	Model	Best	Average
Task-1A		Baseline-random	0.223	0.179
	Task-1A	Baseline-majority	0.0	-
		Baseline-TF-IDF	0.094	-
		Baseline-random	0.567	0.553
TR	₩ Task-1B	Baseline-majority	0.664	-
		Baseline-TF-IDF	0.706	-
		Baseline-random	0.340	0.271
	Task-1C	Baseline-majority	0.0	-
		Baseline-TF-IDF	0.310	-
		Baseline-random	0.148	0.112
Z	Task-1C	Baseline-majority	0.0	-
		Baseline-TF-IDF	0.092	-

3.3.2. Preliminary Results

ARC-NLP-hc: In this part, we share our preliminary classification results for the hand-crafted features: (N)-gram, (H)ashtag, and (U)ser features. We share the classification results for the features both separately and in combination in terms of positive F1 scores in Table 9. As seen

 $^{^{6}} https://gitlab.com/checkthat_lab/clef2022-checkthat-lab/clef2022-checkthat-lab/-/blob/main/task1/baselines/subtask_1.py$

from Table 9, the best positive class F1 score is achieved when the n-gram and hashtag features are combined together. This might be due to the fact that combining hashtags with n-gram features provides a partial solution for contradictory features and cold start problem. However, it is clear from Table 9 that user features alone do not provide any meaningful contribution to classify harmful tweets. Therefore, the model that is trained with all hand-crafted features does not achieve the best score. The best score is also reported in Table 10.

Table 9

Results of our model, ARC-NLP-hc, in terms of positive class F1 score.

	Task	Model	Result
		ARC-NLP-hc-N	0.360
	ARC-NLP-hc-H	0.240	
	≚ Task-1C	ARC-NLP-hc-U	0.090
TR		ARC-NLP-hc-NH	0.400
-		ARC-NLP-hc-NU	0.280
		ARC-NLP-hc-HU	0.120
		ARC-NLP-hc-NHU	0.320

ARC-NLP-pretrain: With the predefined hyperparameters, we implement 10 consecutive training and evaluate the model on the given *Dev_test* set. We use 10 random model initialization. The best (out of 10 runs) and average score of 10 runs are given in Table 10. We observe that ARC-NLP-pretrain produces more robust scores than the BERTurk model. We use 10 random model initializations and report the average score. Although BERTurk achieves better results than ARC-NLP-pretrain, BERTurk's positive class F1 score dip to zero in some of the random initializations in Task-1A and 1C. This results in high deviance between the best and average scores for BERTurk. We argue that a pretrained model using a text corpus without noisy social media texts may not be adequate for such tasks having noisy texts (e.g., emoticons, abbreviations, and web urls) and up-to-date topics (e.g., COVID-19 pandemic). The robustness on noisy texts and the success of our ARC-NLP-pretrain model is promising.

ARC-NLP-contra: After obtaining the best hyperparameters, we implement 10 consecutive training and evaluate the model on the Development Test split. We use 10 random model initialization. The best (out of 10 runs) and average score of 10 runs are given in Table 10. ARC-NLP-contra has better performance in terms of both the best and averaged results in Turkish harmful detection task (1C). ARC-NLP-pretrain has also competitive results with ARC-NLP-contra model but cannot outperform ARC-NLP-contra. Although ARC-NLP-hc exceeds the baseline scores, it obtains lower results compared to previous two models.

3.3.3. Leaderboard Results

Since we obtain the highest positive class F1 score via ARC-NLP-contra method on the given test set, we decide to use its predictions for Task-1C in both Turkish and English. In order to improve our final training before the submission to leaderboard, we utilize the whole labeled

	Task	Model	Best	Average
	Taal 1A	BERTurk	0.400	0.273
	Task-1A	ARC-NLP-pretrain	0.348	0.289
	Task-1B ビ Task-1C	BERTurk	0.794	0.775
К		ARC-NLP-pretrain	0.756	0.745
F		BERTurk	0.572	0.375
		ARC-NLP-hc	0.400	-
		ARC-NLP-pretrain	0.566	0.518
		ARC-NLP-contra	0.600	0.555
Z E	Task-1C	ARC-NLP-contra	0.391	0.323

Preliminary results of our proposed models in terms of the F1 score of positive class for Task-1A and 1C, and the accuracy score for the Task-1B. The highest scores are given in bold.

data by merging the Training, Development, and Development Test splits. We obtain our final model's predictions on the leaderboard Test split. On the other hand, we decide to use ARC-NLP-pretrain for Task-1A and 1B. We only use the Training split for the final training of this model. The list of our submissions and name of the submitted models, along with their leaderboard results, are given in Table 11. Our contradiction check approach for harmful tweet detection, ARC-NLP-contra, takes the first place in the leaderboard.

Since the approach is language-independent, we also submit our winning solution to Task-1C in English. However, it does not perform as expected in English, possibly due to our machine-translated fact list from Turkish to English. We collect COVID-related facts from the Turkish Web Sources (Unicef Turkey and Turkish Ministry of Health). Although some facts are universal (*"Those who are vaccinated do not become infected with HIV"*), the list includes some facts reflected by Turkish politics and cultural effects. An example fact is *"There is no pork ingredient in the inactivated COVID-19 vaccine."* We argue that the performance of our model in English probably decreases because of its fact list that includes direct translations from Turkish fact list. An extension or recollection of fact list for other languages can provide better performances.

Table 11

Leaderboard scores and ranks regarding our submissions in terms of the F1 score of positive class for Task-1A and 1C, and the accuracy score for the Task-1B.

	Task	Submitted Model	Leaderboard Score	Rank
	Task-1A	ARC-NLP-pretrain	0.082	4
TR	Task-1B	ARC-NLP-pretrain	0.760	3
-	Task-1C	ARC-NLP-contra	0.366	1
U E N	Task-1C	ARC-NLP-contra	0.300	6

4. Conclusion

We propose and examine three models for checking worthiness, detecting claims, and detecting harmful tweets in the CLEF2022 CheckThat! Lab. We explain the details of our models and the results of our preliminary experiments in this paper. Our contradiction checking approach, ARC-NLP-contra, is the winning solution for the task of harmful tweet detection in Turkish.

We plan to extend our experiments to other datasets and languages. We can also expand our COVID-related Turkish tweets corpus to train an improved version of ARC-NLP-pretrain. Furthermore, employing a fact list together with Transformer-based language models perform promising as demonstrated by ARC-NLP-contra. The performance can be improved by an extended fact database. We believe that ARC-NLP-contra model can be used for similar tasks requiring an external knowledge base, such as our COVID-19 fact list.

Acknowledgments

We would like to thank the organizers and other participants in the challenge.

References

- P. Nakov, A. Barrón-Cedeño, G. Da San Martino, F. Alam, J. M. Struß, T. Mandl, R. Míguez, T. Caselli, M. Kutlu, W. Zaghouani, C. Li, S. Shaar, G. K. Shahi, H. Mubarak, A. Nikolov, N. Babulkov, Y. S. Kartal, J. Beltrán, The CLEF-2022 CheckThat! Lab on fighting the COVID-19 infodemic and fake news detection, in: M. Hagen, S. Verberne, C. Macdonald, C. Seifert, K. Balog, K. Nørvåg, V. Setty (Eds.), Advances in Information Retrieval, Springer International Publishing, Cham, 2022, pp. 416–428.
- [2] P. Nakov, A. Barrón-Cedeño, G. Da San Martino, F. Alam, J. M. Struß, T. Mandl, R. Míguez, T. Caselli, M. Kutlu, W. Zaghouani, C. Li, S. Shaar, G. K. Shahi, H. Mubarak, A. Nikolov, N. Babulkov, Y. S. Kartal, J. Beltrán, M. Wiegand, M. Siegel, J. Köhler, Overview of the CLEF-2022 CheckThat! Lab on fighting the COVID-19 infodemic and fake news detection, in: A. Barrón-Cedeño, G. Da San Martino, M. Degli Esposti, F. Sebastiani, C. Macdonald, G. Pasi, A. Hanbury, M. Potthast, G. Faggioli, F. Nicola (Eds.), Proceedings of the 13th International Conference of the CLEF Association: Information Access Evaluation meets Multilinguality, Multimodality, and Visualization, CLEF '2022, Bologna, Italy, 2022.
- [3] P. Nakov, A. Barrón-Cedeño, G. Da San Martino, F. Alam, R. Míguez, T. Caselli, M. Kutlu, W. Zaghouani, C. Li, S. Shaar, H. Mubarak, A. Nikolov, Y. S. Kartal, J. Beltrán, Overview of the CLEF-2022 CheckThat! Lab Task 1 on identifying relevant claims in tweets, in: Working Notes of CLEF 2022—Conference and Labs of the Evaluation Forum, CLEF '2022, Bologna, Italy, 2022.
- [4] P. Nakov, G. Da San Martino, F. Alam, S. Shaar, H. Mubarak, N. Babulkov, Overview of the CLEF-2022 CheckThat! Lab Task 2 on detecting previously fact-checked claims, in: Working Notes of CLEF 2022—Conference and Labs of the Evaluation Forum, CLEF '2022, Bologna, Italy, 2022.

- [5] J. Köhler, G. K. Shahi, J. M. Struß, M. Wiegand, M. Siegel, T. Mandl, Overview of the CLEF-2022 CheckThat! Lab Task 3 on fake news detection, in: Working Notes of CLEF 2022–Conference and Labs of the Evaluation Forum, CLEF '2022, Bologna, Italy, 2022.
- [6] D. Q. Nguyen, T. Vu, A. Tuan Nguyen, BERTweet: A pre-trained language model for English tweets, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Online, 2020, pp. 9–14. doi:10.18653/v1/2020.emnlp-demos.2.
- [7] C. Toraman, F. Can, Discovering story chains: A framework based on zigzagged search and news actors, Journal of the Association for Information Science and Technology (2017). doi:10.1002/asi.23885.
- [8] C. Toraman, F. Şahinuç, E. H. Yilmaz, I. B. Akkaya, Understanding social engagements: A comparative analysis of user and text features in Twitter, Social Network Analysis and Mining (2022). doi:10.1007/s13278-022-00872-1.
- [9] T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, ACM, New York, NY, USA, 2016, pp. 785–794. doi:10.1145/2939672.2939785.
- [10] S. Schweter, BERTurk BERT models for Turkish, 2020. doi:10.5281/zenodo.3770924.
- [11] Y. Liu, et al., RoBERTa: A robustly optimized BERT pretraining approach, arXiv preprint arXiv:1907.11692 (2019).
- [12] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186. doi:10.18653/v1/N19-1423.
- [13] C. Toraman, F. Şahinuç, E. H. Yilmaz, BlackLivesMatter 2020: An analysis of deleted and suspended users in Twitter, in: 14th ACM Web Science Conference 2022, WebSci '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 290–295. doi:10. 1145/3501247.3531539.
- [14] C. Toraman, E. H. Yilmaz, F. Şahinuç, O. Ozcelik, Impact of tokenization on language models: An analysis for Turkish, arXiv preprint arXiv:2204.08832 (2022). doi:10.48550/ ARXIV.2204.08832.
- [15] M. Schuster, K. Nakajima, Japanese and Korean voice search, in: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012, pp. 5149–5152. doi:10.1109/ICASSP.2012.6289079.
- [16] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, in: International Conference on Learning Representations, LA, USA, 2019.
- [17] N. Reimers, I. Gurevych, Sentence-BERT: Sentence embeddings using Siamese BERTnetworks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, 2019, pp. 3982–3992. doi:10.18653/v1/D19-1410.
- [18] A. Paszke, et al., PyTorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035.

- T. Wolf, et al., Transformers: State-of-the-art natural language processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Online, 2020, pp. 38–45. doi:10.18653/v1/2020.emn1p-demos.6.
- [20] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperparameter optimization framework, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 2623–2631. doi:10.1145/3292500.3330701.