NLP&IR@UNED at CheckThat! 2020: A Preliminary Approach for Check-Worthiness and Claim Retrieval Tasks using Neural Networks and Graphs ⋆

Juan R. Martinez-Rico[1], Lourdes Araujo[1,2], and Juan Martinez-Romo[1,2]

[1] NLP & IR Group, Dpto. Lenguajes y Sistemas Informáticos
Universidad Nacional de Educación a Distancia (UNED), Madrid 28040, Spain
jrmartinezrico@invi.uned.es
[2] Instituto Mixto de Investigación - Escuela Nacional de Sanidad (IMIENS)
lurdes@lsi.uned.es, juaner@lsi.uned.es

Abstract. Check-Worthiness and Claim Retrieval are two of the first tasks to be performed on the Fake News detection pipeline. In this article we present our approach to these tasks presented in the 2020 edition of the CheckThat! Lab. In the task 1, Tweet Check-Worthiness English, we propose a Bi-LSTM model with Glove Twitter embeddings where the number of inputs has been increased with a graph generated from the additional information provided for each tweet. In task 1 Arabic we have followed a similar approach but using a feed forward neural network model with Arabic embeddings. For the task 5, Debate Check-Worthiness, we propose a naive Bi-LSTM model with Glove embeddings. Finally, our approach to the task 2, Claim Retrieval, is based in a feed forward neural network model with features such as cosine similarity over Universal Sentence Encoder embeddings of tweets and claims, and other linguistic features extracted from both elements.

Keywords: check-worthiness · claim retrieval · embeddings · graph features

1 Introduction

One of the problems that current society faces is the spreading of fake news. The influence of this type of news in the results of the United States presidential elections a few years ago, and more recently, the disinformation that they are

⋆ This work has been partially supported by the Spanish Ministry of Science and Innovation within the projects PROSA-MED (TIN2016-77820-C3-2-R), DOTT-HEALTH (PID2019-106942RB-C32) and EXTRAE II (IMIENS 2019).

causing in the current pandemic of COVID2019 are two examples of this phenomenon. To combat this problem, sites dedicated to checking the veracity of the news that circulate daily in traditional media and on social networks have proliferated. These sites normally carry out their work through human experts who review the news in circulation every day. To facilitate the work of these experts, some systems that prioritize the claims to be verified or return a list of claims that verify another given as input have been proposed. On the other hand, these two tasks can also be part of a larger system of claim verification and detection of fake news.

This article describes the participation of the NLP&IR@UNED[3] team in tasks T1, T2 and T5 of the CheckThat! Lab at CLEF2020[1][2]. Tasks T1 and T5 are dedicated to prioritizing claims to be verified (check-worthiness) and task T2 is a claim retrieval task.

The rest of the article is organized as follows: in section 2 we describe the different approaches that we have tried to face each task, in section 3 we discuss the results we have obtained, and section 4 is devoted to reflect our conclusions and future work.

2 Proposed Approaches

2.1 Task 1 - Tweet Check-Worthiness in English and Arabic

The purpose of this task is, given a stream of tweets and one or more topics, sort the tweets according to their check-worthiness for the topic to which they are supposed to belong. Three topics have been provided for the Arabic language: “The protests in Lebanon”, “Wassim Youssef” and “Turkey enters Syria”, and all English language tweets belong to topic “COVID19”. The official evaluation measure for Arabic language is P@30 and for English language is MAP.

To address these two versions of the task 1, five different models have been analyzed. All models use cross entropy as a loss function and accuracy as a metric[4].

Model 1 The first of these models[5] is a feed forward neural network (FFNN) whose input is made up of word embeddings of the first n words of the tweet text, followed by a 1D global max pooling layer that operates on the n word vectors, a hidden layer, and a sigmoid final layer of size 1 that provides the check-worthiness score between 0 and 1.

[3] Identified as NLPIR01 in the official results.
[4] We plan to release the source code at https://github.com/jrmtnez/NLP-IR-UNED-at-CheckThat-2020.
[5] All models have been implemented in Tensorflow 2.1 with Keras: https://www.tensorflow.org

Embedding features Word embeddings can be self-generated during training, or they can be preloaded at startup. For this second option, English pretrained Twitter Glove[3] embeddings of dimension 200[6] and Glove Arabic embeddings of dimension 256[7] have been used respectively in each version of task 1.

Additionally, in the Arabic version of task 1, the title and the description of the topic to which each tweet belongs have been concatenated to the text of the tweet to form the model input.

To preprocess the raw Arabic text, we use as tokenizer the *simple_word_tokenize* from Camel Tools[4]. The first 25 tokens of each tweet and the first 100 tokens of the topic title and topic description concatenation have been selected to form an input of size 125.

For the English language, the NLTK[8][5] tokenizer has been used, and the first 50 tokens have been selected as input.

Graph features Taking advantage of the fact that the organizers have provided the complete information of each tweet in a json file, we have implemented a process to increase the size of the input with tweets related to the current tweet. To do this, from the information of the tweets contained in the training, *dev* and *test* datasets, we extract triples of type *tweet-hashtags-hashtag*, *tweet-quoted-tweet*, *tweet-reply_status-tweet*, *tweet-contain_url-url* and *tweet-has_mention-user*, and build a graph per dataset with these triples.

After this, for each tweet present in the datasets, we search for the first three tweets that are neighbors of the current one, for example, we would go from the *tweet* node to a *hashtag* node and from this we would select three *tweet* nodes. If there were no *hashtag* neighbor nodes or the *hashtag* neighbor nodes did not have three or more related *tweet* nodes, from the initial *tweet* node we search for *tweet* nodes behind relations *quoted*, *reply_status*, *contain_url* and *has_mention* in this order. The result is that the texts of up to three tweets can be concatenated to the text of the original tweet.

As we will see later in section 3.1, the model can be executed indistinctly with the inputs from a single instance, or with the inputs concatenated from several instances if we make use of the tweets graph.

Model 2 The second model is a CNN fed by the same features as Model 1. The input and embedding layers are followed by two pairs of convolutional 1D and max pooling 1D layers, a flatten layer that feeds a dense layer, and finally a dense sigmoid layer of size 1 at the output. Each convolutional 1D layer has a kernel size of 5 and the max pooling 1D layers have a pool size of 5.

Model 3 The third model is a LSTM network fed by the same features as Model 1. In this case, after the input and embedding layers there is a LSTM layer that feds directly the dense sigmoid layer of size 1 that forms the output.

[6] https://nlp.stanford.edu/projects/glove/
[7] https://github.com/tarekeldeeb/GloVe-Arabic
[8] https://www.nltk.org/

Model 4 The fourth model is a Bi-LSTM network fed by the same features that previous models. After the input and embedding layers there are two bidirectional LSTM layers followed by a dense layer that precedes the output sigmoid layer.

Model 5 The last model is again a FFNN but in this case the input is made up of tf-idf vectors. To build this input we have followed the same strategy that in the embedding and graph features of previous models: in Arabic language the title and the description of the topic have been concatenated to the twitter text, and up to three additional tweets have been added to the input from the graph extracted from the tweet information.

The network is made up of three pairs of dense and batch normalization layers and the size of the dense layers decreases by 50% in each stage. These six layers are followed by a batch normalization layer, a dropout layer, and finally a sigmoid output layer.

2.2 Task 5 - Debate Check-Worthiness in English

The objective of this task is, given a transcripted political debate segmented into sentences and with the speakers annotated, sort the sentences according to their check-worthiness. The official evaluation measure for this task is MAP.

In this task, the five models described in section 2.1 have been used with slight variations. First, FFNN models have also been run without a hidden layer. On the other hand, in addition to word embeddings and tf-idf vectors, another type of input data has been used in this model as we explain below.

To perform a text analysis of the sentences we have prepared a version of the English Regressive Imagery Dictionary (RID)[6][7] with a format compatible with that used by *liwc* python module[9]. This dictionary contains 3150 words and roots in 48 categories, and these in turn are grouped into three main categories: *primary*, *secondary* and *emotion*.

The input vector consists of 51 decimal numbers, one for each category. For each instance of the *training* and *test* datasets, we calculate the percentage of words that are in those 51 categories.

2.3 Task 2 - Claim Retrieval in English

In this task, for each check-worthy tweet provided, a ranked list of claims must be returned, reflecting which claims best support that tweet. The official evaluation measure for this task is MAP@5.

In our approach we have used an FFNN similar to model 5 described above, and for this model we build a dataset in the following way: for each claim we calculate its sentence embedding v_c with Universal Sentence Encoder[8], the ratio of different tokens to total tokens rt_c, the average number of characters

[9] https://pypi.org/project/liwc/

per word avc_c, the number of verbs vn_c, the number of nouns nn_c, the ratio of content words[10] rcw_c regarding the total number of words, and the ratio of content tags[11] rct_c with respect to the total of tags.

The same values v_t, rt_t, avc_t, vn_t, nn_t, rcw_t, rct_t are calculated for each tweet, and for each claim title v_{ct}, rt_{ct}, avc_{ct}, vn_{ct}, nn_{ct}, rcw_{ct}, rct_{ct}.

Combining claims and tweets and claim titles and tweets we obtain the features for our dataset shown in table 1.

Claim - Tweet	**Claim title - Tweet**
$sim_{ct} = cosine\_sim(v_c,\ v_t)$	$sim_{ctt} = cosine\_sim(v_{ct},\ v_t)$
$drt_{ct} = rt_c\text{-}rt_t$	$dct_{ctt} = rt_{ct}\text{-}rt_t$
$davc_{ct} = avc_c\text{-}avc_t$	$davc_{ctt} = avc_{ct}\text{-}avc_t$
$dvn_{ct} = vn_c\text{-}vn_t$	$dvn_{ctt} = vn_{ct}\text{-}vn_t$
$dnn_{ct} = nn_c\text{-}nn_t$	$dnn_{ctt} = nn_{ct}\text{-}nn_t$
$drcw_{ct} = rcw_c\text{-}rcw_t$	$drct_{ctt} = rct_{ct}\text{-}rct_t$
$drct_{ct} = rct_c\text{-}rct_t$	$drcw_{ctt} = rcw_{ct}\text{-}rcw_t$

Table 1: Task 2 features

3 Experiments and Results

3.1 Task 1 - Tweet Check-Worthiness in English and Arabic

In task 1, for both, English and Arabic languages, we have increased the size of the input by relying on the creation of a graph with which we retrieve tweets related to each instance of the datasets based on the hypothesis that the relationships *tweet-hashtags-hashtag*, *tweet-quoted-tweet*, *tweet-reply_status-tweet*, *tweet-contain_url-url* and *tweet-has_mention-user* can enrich the information provided to the different classifiers. To verify this, we have performed a grid search with different parameters that were applied or not according to the model used. These parameters have been: graph generated inputs, pretrained embeddings, number of epochs, batch size, hidden layer size, activation type, optimizer type, dropout, number of epochs and batch size. All models use the adam optimizer with its default parameter values except the model 4 which uses nadam.

In Arabic there was no *dev* dataset so we extracted 20% of the *training* dataset instances as a *dev* dataset.

[10] Nouns, verbs, adjectives and adverbs.
[11] "NN", "NNS", "NNP", "NNPS", "VB", "VBD", "VBG", "VBN", "VBP", "VBZ", "JJ", "JJR", "JJS", "RB", "RBR", "RBS" and "WRB"

Table 2 shows the best results obtained for the different combinations of the use of pretrained embeddings and graph features and the optimal parameters for each model in Arabic language on the *dev* dataset, and table 3 shows best results for English language.

Model	Graph	Emb.	Activation	H. layer	Dropout	Ep./Bch.	MAP
Model 1	N	N	relu	2000	-	50/8	0.1175
(FFNN)	N	Y	relu	2000	-	50/8	0.1399
	Y	N	relu	2000	-	50/8	0.1238
	Y	**Y**	**relu**	**2000**	**-**	**50/8**	**0.1454**
Model 2	N	N	relu	100	-	25/8	0.1201
(CNN)	N	Y	relu	100	-	25/8	0.1436
	Y	N	relu	100	-	25/8	0.1225
	Y	Y	relu	100	-	25/8	0.1313
Model 3	N	N	tanh	-	0.1	5/32	0.0646
(LSTM)	N	Y	tanh	-	0.1	5/32	0.0821
	Y	N	tanh	-	0.1	5/32	0.0634
	Y	Y	tanh	-	0.1	5/32	0.0797
Model 4	N	N	tanh	10	0.1	5/32	0.1145
(Bi-LSTM)	N	Y	tanh	10	0.1	5/32	0.1131
	Y	N	tanh	10	0.1	5/32	0.1211
	Y	Y	tanh	10	0.1	5/32	0.1210
Model 5	N	-	relu	500	0.4	10/8	0.0665
(FFNN TF/IDF)	Y	-	relu	500	0.4	10/8	0.0640

Table 2: Task 1 - Arabic parameter analysis

After observing the results obtained for the Arabic language, we can see that in almost all cases the use of Glove Arabic embeddings improves the result, and that the expansion of inputs through the tweet graph improves five of the nine possible configurations, one of them (FFNN + Graphs + Embedings) being the one that obtains the best global result.

For the English language again it is confirmed that the use of Glove embeddings improve performance in all cases. The graph features do not show a homogeneous behavior, although the best value is obtained for the Bi-LSTM model with embeddings and graph features.

Regarding the official results, in Arabic language our best run was the Bi-LSTM model with Glove Arabic embeddings and graph features, sent as *contrastive2* which obtained a P@30 of 0.5333, ranking 13th out of 28 runs sent by all the teams, while the FFNN (*primary*) and CNN (*contrastive1*) models with the same features obtained a P@30 of 0.3917 (ranking 19th) and 0.4833 (ranking 16th) respectively. In English language our best run was the Bi-LSTM model (*primary*) which obtained a MAP of 0.6069 (ranking 20th) and the FFNN (*con-*

Model	**Graph**	**Emb.**	**Activation**	**H. layer**	**Dropout**	**Ep./Bch.**	**MAP**
Model 1	N	N	hard_sigmoid	1000	-	100/8	0.6480
(FFNN)	N	Y	hard_sigmoid	1000	-	100/8	0.7145
	Y	N	hard_sigmoid	1000	-	100/8	0.6097
	Y	Y	hard_sigmoid	1000	-	100/8	0.7214
Model 2	N	N	sigmoid	32	-	10/16	0.5392
(CNN)	N	Y	sigmoid	32	-	10/16	0.6671
	Y	N	sigmoid	32	-	10/16	0.5998
	Y	Y	sigmoid	32	-	10/16	0.6507
Model 3	N	N	tanh	-	0.1	5/8	0.5578
(LSTM)	N	Y	tanh	-	0.1	5/8	0.6067
	Y	N	tanh	-	0.1	5/8	0.4153
	Y	Y	tanh	-	0.1	5/8	0.4385
Model 4	N	N	tanh	10	0.1	10/64	0.6188
(Bi-LSTM)	N	Y	tanh	10	0.1	10/64	0.7410
	Y	N	tanh	10	0.1	10/64	0.6024
	Y	**Y**	**tanh**	**10**	**0.1**	**10/64**	**0.7425**
Model 5	N	-	sigmoid	1000	0.4	50/8	0.4695
(FFNN TF/IDF)	Y	-	sigmoid	1000	0.4	50/8	0.4634

Table 3: Task 1 - English parameter analysis

trastive1) and CNN (*contrastive2*) models obtained a MAP of 0.5546 (ranking 22th) and 0.5193 (ranking 23th), respectively.

3.2 Task 5 - Debate Check-Worthiness in English

In task 5, different parameters and settings have also been analyzed. In this case, a *dev* dataset was not available either, so we have partitioned the *training* dataset leaving 20% of it as a *dev* dataset. Given the great imbalance of classes (4027 negative instances and 42 positive instances) we have opted for an oversampling strategy to equalize the number of positive and negative instances. In this task, an additional FFNN model 6 that makes use of features derived from a RID text analysis has been used.

Table 4 shows the best results obtained for the different combinations of the use of pretrained embeddings and oversampling, and the optimal parameters for each model. All models use the adam optimizer with its default parameter values.

As we can see, the use of oversampling in general does not improve the behavior of the different models. The best results in FFNN models are obtained with the use of RID-based features without oversampling, outperforming FFNN models with inputs based on embeddings and TF/IDF vectors. It is also clearly seen, as was the case in task 1, that the use of 6B-100D Glove pretrained embeddings substantially improve the performance of all the models in which it can be used.

The model that outperforms the rest by far is the Bi-LSTM with Glove embeddings. All of the runs that we submitted for task 5 were based on this model. The *contrastive2* run used oversampling, while the *primary* and *contrastive1* runs did not use oversampling, and shared the same parameters. The only difference between them was a different random weight initialization.

Model	**Over.**	**Emb.**	**Activation**	**H. layer**	**Dropout**	**Ep./Bch.**	**MAP**
Model 1	N	N	sigmoid	0	-	50/8	0.0482
(FFNN)	N	Y	sigmoid	0	-	50/8	0.0605
	Y	N	sigmoid	0	-	50/8	0.0410
	Y	Y	sigmoid	0	-	50/8	0.0511
	N	N	sigmoid	1000	-	20/64	0.0538
	N	Y	sigmoid	1000	-	20/64	0.0547
	Y	N	sigmoid	1000	-	20/64	0.0413
	Y	Y	sigmoid	1000	-	20/64	0.0760
Model 2	N	N	sigmoid	32	-	20/64	0.0374
(CNN)	N	Y	sigmoid	32	-	20/64	0.0808
	Y	N	sigmoid	32	-	20/64	0.0128
	Y	Y	sigmoid	32	-	20/64	0.0282
Model 3	N	N	tanh	-	0.2	20/64	0.0375
(LSTM)	N	Y	tanh	-	0.2	20/64	0.0768
	Y	N	tanh	-	0.2	20/64	0.0374
	Y	Y	tanh	-	0.2	20/64	0.0487
Model 4	N	N	tanh	256	0.2	10/16	0.0239
(Bi-LSTM)	**N**	**Y**	**tanh**	**256**	**0.2**	**10/16**	**0.1700**
	Y	N	tanh	256	0.2	10/16	0.0256
	Y	Y	tanh	256	0.2	10/16	0.1050
Model 5	N	-	sigmoid	1000	0.4	20/64	0.0719
(FFNN TF/IDF)	Y	-	sigmoid	1000	0.4	20/64	0.0421
Model 6	N	-	elu	102	0.4	20/64	0.0871
(FFNN RID)	Y	-	elu	102	0.4	20/64	0.0268

Table 4: Task 5 - English parameter analysis

Table 5 shows the official results of this task. Our *primary* and *contrastive1* runs based on the Bi-LSTM model with Glove embeddings obtained the first positions in the classification.

Team	Run	MAP
NLPIR01	**Primary**	**0.0867**
NLPIR01	**Contrastive-1**	**0.0849**
UAICS	Primary	0.0515
UAICS	Contrastive-1	0.0431
TobbEtuP	Contrastive-1	0.0417
NLPIR01	**Contrastive-2**	**0.0408**
UAICS	Contrastive-2	0.0328
TobbEtuP	Primary	0.0183

Table 5: Task 5 - Official results

3.3 Task 2 - Claim Retrieval in English

In this task we have used an FFNN with 1000 elu[12] units in the first hidden layer, 500 elu units in the second hidden layer and 250 elu units in the last hidden layer, and Universal Sentence Encoder embeddings of tweets, claims and claim titles. We have used the adam optimizer with its default values and we have trained the model for 50 epochs with a batch size of 128.

In our experiments on the *dev* dataset we were able to verify that the use of the features derived from the claim title do not provide improvements in the performance offered by the claim-tweet features. Table 6 shows the results obtained with both configurations. Both sets of features exceed the MAP@5 of 0.609 obtained by the baseline based on Elasticsearch provided by the organization.

We have submitted three runs with two different settings. In the first one, sent as *primary*, we have made use of the seven features described in section 2.3 involving claims and tweets. With this configuration we obtained a MAP@5 of 0.8560, placing our team in fourth place.

In the *contrastive1* run we used all features and the *contrastive2* was identical to the *primary* run but with a different random initialization. With these two configurations we obtained respectively a MAP@5 of 0.8390 and 0.8550, confirming that the seven *Claim title - Tweet* features do not improve performance when used together with the seven *Claim - Tweet* features.

Feature Set	MAP 1	MAP 3	MAP 5	MAP 10	MAP 20	MAP All
Claim-Tweet	0.739	0.793	0.798	0.802	0.804	0.805
All	0.723	0.781	0.787	0.790	0.792	0.794
Baseline	0.470	0.601	0.609	0.615	0.617	0.619

Table 6: Task 2 - Feature set comparison on *dev* dataset

[12] Exponential linear unit.

4 Conclusions and Future Work

In this paper, we present our approximation to the tasks *tweet check-worthiness*, *debate check-worthiness* and *claim retrieval* at the CLEF-2020 Check-That! Lab.

Examining the results obtained in the two check-worthiness tasks to which we have participated, three if we take into account that the first was defined in two different languages, we can see that the use of pretrained embeddings of the appropriate language, significantly improves the performance of the models with respect to generating these embeddings during the training phase.

In the two versions of task 1, we have used a particular method to increase the information that reaches the input of the models, using a graph constructed from the tweets provided in the datasets that collects the relationships *tweet-hashtags-hashtag*, *tweet-quoted-tweet*, *tweet-reply_status-tweet*, *tweet-contain_url-url* and *tweet-has_mention-user*, between tweets. Although this mechanism has not behaved in a homogeneous way throughout the different models, it has been the one that has obtained the highest MAP values in the *dev* dataset in both the Arabic and English versions of this task.

In task 5 we have made use of RID-based features and, although we have not sent any run with them, in our tests with FFNN models on the *dev* dataset, we have obtained good results compared to embeddings and tf-idf based features, so this type of text analysis-based features can be an alternative to more well-known ones like LIWC[9]. We did not have these features prepared in time for task 1 but we think they can be applied to tweet texts and it will be a job to be done in the future.

MAP values in this task are certainly low. We think that the large class imbalance and the small number of positive instances can cause these instances to not be correctly characterized by the models.

In the claim retrieval task we have assumed that the similarity between claim, claim title and tweet would allow a selection of claims appropriate to the requirements of this task and for this, we have implemented a mixed strategy using sentence embeddings and stylometric features based on token counts and ratios, far exceeding the provided baseline with these features.

On the other hand, the balance of our participation in these tasks has been positive, obtaining first place in task 5, fourth in task 2 and more discreet results in task 1.

In future work we plan to continue experimenting with graph features to increase the size of the inputs or the number of training instances, combining this strategy with more sophisticated language representations such as BERT[10].

References

1. Alberto Barron-Cedeno, Tamer Elsayed, Preslav Nakov, Giovanni Da San Martino, Maram Hasanain, Reem Suwaileh, Fatima Haouari, Nikolay Babulkov, Bayan Hamdan, Alex Nikolov, Shaden Shaar, and Zien Sheikh Ali. Overview of CheckThat! 2020: Automatic Identification and Verification of Claims in Social Media. *arXiv:2007.07997 [cs]*, July 2020. arXiv: 2007.07997.

2. Alberto Barrón-Cedeño, Tamer Elsayed, Preslav Nakov, Giovanni Da San Martino, Maram Hasanain, Reem Suwaileh, and Fatima Haouari. CheckThat! at CLEF 2020: Enabling the Automatic Identification and Verification of Claims in Social Media. In Joemon M. Jose, Emine Yilmaz, João Magalhães, Pablo Castells, Nicola Ferro, Mário J. Silva, and Flávio Martins, editors, *Advances in Information Retrieval*, Lecture Notes in Computer Science, pages 499–507, Cham, 2020. Springer International Publishing.
3. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.
4. Ossama Obeid, Nasser Zalmout, Salam Khalifa, Dima Taji, Mai Oudah, Bashar Alhafni, Go Inoue, Fadhl Eryani, Alexander Erdmann, and Nizar Habash. CAMeL Tools: An Open Source Python Toolkit for Arabic Natural Language Processing. page 11, 2020.
5. Edward Loper and Steven Bird. NLTK: The Natural Language Toolkit. *arXiv:cs/0205028*, May 2002. arXiv: cs/0205028.
6. Colin Martindale. Romantic Progression: The Psychology of Literary History, Hemisphere, Washington, DC, 1975. *Google Scholar*, 1975.
7. Colin Martindale. *The clockwork muse: The predictability of artistic change.* The clockwork muse: The predictability of artistic change. Basic Books, New York, NY, US, 1990. Pages: xiv, 411.
8. Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, and Chris Tar. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.
9. James W. Pennebaker, Ryan L. Boyd, Kayla Jordan, and Kate Blackburn. The development and psychometric properties of LIWC2015. Technical report, 2015.
10. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.