Classification of Relevant Comments from Competitive Programming Discussions

***Abstract*—In competitive programming, understanding a problem often requires more than just the official solution. Users typically turn to comments in the contest's thread for additional insights. These comments often contain irrelevant information, necessitating manual identification of relevant ones. This paper introduces CommentThreadFilter, a system designed to classify comments as either Relevant or Irrelevant with respect to the main thread post. Leveraging base models like *BERT*, *RoBERTa* and *SciBERT*, we evaluate the system's performance on the newly created CFComments dataset. The dataset is the first of its kind, comprising 19 labelled comment threads in the competitive programming domain, manually annotated by two experts, alongside 1131 unlabeled comment threads. The proposed models, combined with a weak augmentation on the text, achieve an F1-score of 86%, outperforming the 77% F1-score obtained using gpt-3.5-turbo. By effectively filtering comments as Relevant or Irrelevant, our system enhances the user's ability to gain valuable insights and better comprehend the underlying problem.**

***Index Terms*—comment threads, relevant comment detection, thread-based transformer, competitive programming**

I. Introduction

Competitive programming is a mind sport in which various participants try to come up with solutions for given algorithmic programming problems by respecting several constraints. In general, a competitive programming problem consists of the *problem statement*, which contains the description of the problem along with the input constraints (e.g. *Find if a number* x *is prime,* $x \leq 10^6$.) and *input/output* examples (e.g. *Input:* 23 *Output:* $prime$*, Input:* 24 *Output:* $not\ prime$.)

Participants in contests need to implement an algorithmic solution that passes all the test cases (input and output pairs) and fits within the memory and time constraints. An online judge (e.g., Codeforces[1]) is a website that hosts problems in which users can participate in different contests. Codeforces hosts two main types of contests: Educational and Division (Div). The Educational contests consist of problems that present a classic idea, while the Div contests usually have more ad-hoc problems. Usually, a round consists of 5-7 problems typically denoted with capital letters (e.g. A-G or more). A given problem has an official blog attached to it where an official editorial and solution is published. The discussions that users tend to have in the blogs are either related to an official solution (questions, more detailed explanations), hints to other users who do not understand a given problem, questions related to their solution on why it fails (a bug, an incorrect idea etc.) and *irrelevant* comments, which are unrelated to competitive programming in general.

[1] https://codeforces.com, last accessed on 1st February 2024

In this work, we investigate the predictability of relevant and irrelevant comments in competitive programming by using models from the BERT [4] family. We create a dataset comprising comments from educational and div rounds, containing nine types of comments that provide educational value for a competitive programmer, along with examples of irrelevant comments. Due to the dataset's limited and unbalanced nature, we group the nine types as a single label, transforming the problem into a binary classification task. Additionally, we compare with a Generative AI system, such as ChatGPT, revealing that smaller transformer models (e.g. BERT) yield better results. These models are either finetuned or not on the Masked Language Modelling (MLM) task on the unlabelled dataset. During the training process on the supervised dataset, all the weights are frozen except the head. Weak augmentations are also employed to reduce the risk of overfitting. Since other papers have used the comment tree as part of the features, we also propose a method to tackle this issue, but we find that the results are very similar to the case of not using the tree. This is because we don't use the statements or editorials as part of the comment tree, due to the length of the text. A model must also learn to associate the comment with the correct part of the text that represents the actual statement and editorial.

II. Related Work

The problem of finding relevant comments has also been studied in different domains. For example, Cho et al. [2] train a classifier that determines if the comments are helpful in the case of peer review documents in physics. In this case, the comments were created by 44 reviewers, who are undergraduate students, in a controlled environment. In contrast, our investigation deals with comments written by any user, leading to a higher variance. Additionally, in competitive programming, an idea can be distributed across multiple comments, as users may engage in a conversation about a specific solution, incrementally providing different pieces of information.

Want et al. [15] use an unsupervised LDA-based approach to identify diversionary comments in political blogs, emphasizing the significance of the comment tree structure in determining the type of comment. We tried to use the comment tree as part of the workflow by employing an attention mechanism, but this didn't bring an improvement over using just plain comments. This could be attributed to our decision not to treat editorial and problem statements as top comments due to their lengthy nature, requiring models to learn complex mappings.

The extraction of relevant comments has also been examined on YouTube comments [5, 11], showing that relevant comments extracted with various text similarity techniques are perceived as useful by users.

III. CFComments Dataset

Critical data from Online Judges (e.g., Codeforces) includes **(1) Statements, (2) Official Editorial (Natural Language Description + Code), (3) Input / Output, (4) User submission and (5) Blog content**.

While most datasets cover (1), (2), (3) and (4), (5) is often overlooked [9]. Blog content, typically in the form of comments, provides additional information related to (1), (2), (3), and (4), with users seeking directions, alternative solutions, clarifications, hints for bug fixes, or better understanding of the problem.

To address (5), we introduce the **CFComments** dataset, comprising a **labelled dataset** and an **unlabelled dataset**. The labelled dataset includes 16 Educational blog round comments and 3 Div blog round comments. The unlabelled dataset encompasses all Educational and Div rounds from Codeforces, including labelled ones, totaling 1131 blog rounds.

Each comment in our dataset has the following attributes:

- *Uri*: URI of the comment.
- *Id*: Id of the comment.
- *Father_id*: Father's comment id of the comment in the comment thread or -1 if the comment is a root.
- *Username*: Relative path to the profile of the comment's user.
- *Text*: Comment text.
- *Comment_rating*: The rating of the comment.
- *Problem*: The problem the comment relates to (A, B, C, etc.) or Irrelevant if it can't be matched to a given problem. The comment does not need to explicitly specify the problem, as it can be derived from the conversation.
- *Label*: The label of the comment.
- *Timestamp*: Timestamp of the comment creation.
- *Round_id*: The round id the blog comment relates to.
- *Round_name*: The round name the comment relates to.

Each comment has been labelled by manual annotators using one of the following classes:

- *SolutionExp*: A detailed comment explaining how to solve a problem, more like a step-by-step guide.
- *HintExp*: It can be viewed as a sparse variant of SolutionExp in which various elements from an entire solution are presented.
- *AlgoExp*: A comment which states the algorithm used and the submission link with little to no information.
- *TimeComExp*: A comment which states the complexity and the submission link with little to no information.
- *FixingExp*: A comment which explains or hints how to solve a specific bug in an implementation
- *TestExp*: A comment which explains the result of a test case.
- *QuestionExp*: A relevant question about a specific problem in the contest. It should have a relevant answer in the comment subtree.
- *SubQuestion*: A comment which asks insights about why a submission fails.
- *TestQuestion*: A comment asking for insights about a specific test's result.
- *Irrelevant*: Any other comment that does not fit the above labels.

In this paper, we reframe all the comments different from the *Irrelevant* label as *Relevant* because the distribution of these labels is very unbalanced. Some examples from the dataset can be viewed in Table I. While the paper reports only the binary classification task, the released data and code covers the multi-class scenario as well.

A. Inter-rater Reliability

Two graduate students with experience in Competitive Programming contests labelled a subset of 4 Educational rounds to measure how hard predicting would be. To calculate the inter-rater reliability, **Cohen's Kappa** [3] was used.

In Table II, Cohen's Kappa score is **0.84**, which implies that agreement is strong.

	Irrelevant	Relevant	Row Total
Irrelevant	78	4	82
Relevant	12	127	139
Column Total	90	131	221
Cohen's Kappa	-	-	0.84

TABLE II
INTER-AGREEMENT MATRIX REFRAMED

B. Exploratory Data Analysis

The dataset has $\sim$ 900 div rounds and $\sim$ 150 educational rounds. The ratio of *Irrelevant* comments is much higher in the Div fold than in the Educational fold. More users are engaged in the Div contests since those contests are usually rated, while the Educational rounds are not. Another observation is that the labels are unbalanced, and since the labelled dataset is small, the resulting model would have poor results.

In Figure 1 it can be seen the distribution of the labels. Div contests seem harder to predict and contain more noisy data. The figures are based on unlabelled and labelled rounds.

IV. Proposed approach - CommentThreadFilter

To classify the comments as *Relevant* and *Irrelevant*, we preprocessed the text, applied various weak augmentation steps, and transformed it into a numerical representation using different models of the BERT family. Since the dataset is minimal, we chose not to train the entire network but to train the weights of the attached head. We also tried to use the ancestors of each comment by employing an attention mechanism, but the results didn't improve. The method is described below.

Text	Label	Round_name
There are two different k in the editorial of E.	Irrelevant	EDU CF Round 125
Wow, such a fast tutorial!	Irrelevant	DIV CF Round 816
My approach for problem C, Binary search the waiting time before starting to move. <a href="https://codeforces.com/contest/1716/submission/166993379"$)$https://codeforces.com/contest/1716/submission/166993379	Relevant (AlgoExp)	EDU CF Round 133
You got some misunderstanding on this problem. The correct process is that 1+floor(1/1)=2 2+floor(2/2)=3 3+floor(3/1)=6 6+floor(6/6)=7 Then 7 can be made in 4 moves.	Relevant (TestExp)	EDU CF Round 122
Array.sort() worst complexity is O(n'2) instead use collection.sort()	Relevant (FixingExp)	EDU CF Round 130
Oh I get it. We can use linear sieve to find all primes p and compute p^i. For a composite $x = a \cdot b$, compute x^i as $a^i \cdot b^i$. This is $O(k \log i / \log k) = O(i)$	Relevant (HintExp)	EDU CF Round 123
Could you explain the idea of your code?	Relevant (QuestionExp)	EDU CF Round 123
Solution to the first problem: If the number of units is 0 then the answer is 0, otherwise if the number of units is 4 then the answer is 2, otherwise the answer is 1.	Relevant (SolutionExp)	EDU CF Round 131

TABLE I
EXAMPLES OF COMMENTS FOR EACH LABEL IN THE DATASET

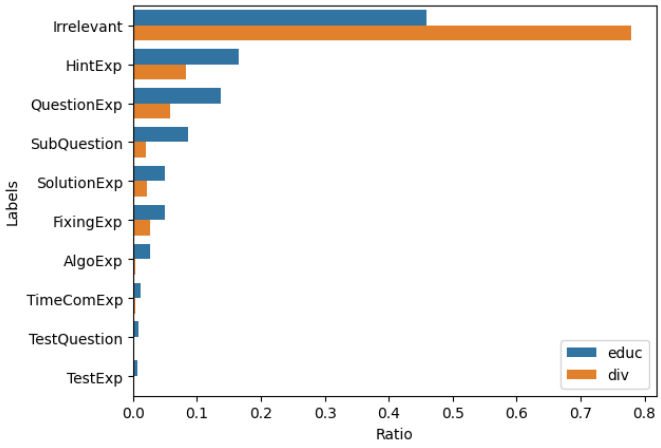


Fig. 1. Label distribution labelled fold

A. Preprocessing

To reduce the length of the text without losing too much information, we applied the following preprocessing:

- Removed all the HTML markups and replaced the content of the *<code>* tag with the string *"(code)"*. This offers the possibility of more extended natural language contexts since having the entire code would impose an earlier truncation of the text.
- Replaced the submissions links with the string *"(link to problem {A, B, C,..})"*. Some labels have a higher frequency of submission links than others. For example, a *SolutionExp* usually also has a submission link.
- Reframed the task of predicting all the labels to predict if a comment is *Relevant* or *Irrelevant*. All the labels except the *Irrelevant* were transformed into the *Relevant*.

1) Modelling: We have a dataset of comments C and a thread T that specifies the connections between the observations as a tree.

We define $T = \{(c_t, c_i) | indegree(c_t) \leq 1, indegree(c_i) \leq 1, c_t, c_i \in C\}$ as the thread.

In our dataset, a thread is represented by multiple disjoint trees, hence the -1 label for the father_id. One can connect all the disjoint trees to a virtual node or consider the main blog post as a node and connect to it. We consider the first option because the text size is considerably longer in the blog post. In many cases, it is necessary to split it into chunks to create embeddings. Besides this, the post contains the editorial for all the problems in the round. Determining which problem a given comment refers to is not easy since the Irrelevant comments may refer or not to any problem.

We define a k-ancestor context $ancestor_{context}(c, k)$ as the comments from the path between a comment $c \in C$ and it's k-th ancestor $c_k \in C$:

$ancestor_{context}(c, k) = \{c_j | 0 < level(c_j) - level(c_i) \leq k, c_j \ ancestor \ of \ c_i\}$

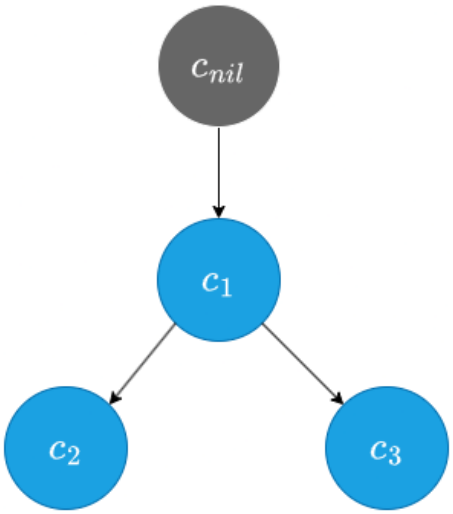
If there is no k-th ancestor comment, we attach a virtual comment named c_{nil}.

We denote the representation of a comment c as $ancestor_{context}(c, k) \cup c$. An example can be seen in Figure 2 To transform c and its ancestors in a numerical form, a pre-trained transformer Tr can be used. Given a transformer Tr with hidden size h, l transformer blocks and a hidden representation of the classification token $\vec{cls}^{(i)}$ in layer $i, 1 \leq i \leq l$, the hidden representation of a comment c is $\vec{c} = \frac{1}{p} \sum_{i=l-p+1}^{l} \vec{cls}^{(i)}, \vec{c} \in \mathbf{R}^h, \vec{cls}^{(i)} \in \mathbf{R}^h$. In our case, we use the three layers.

Let us consider that a comment c can be represented as $(Ac, \vec{c}, \vec{mask})$, where $Ac \in \mathbf{R}^{kxh}$ is a matrix which contains the hidden representations of the comments in the $ancestor_{context}$ where the comments are ordered by their level in the thread, $\vec{c}$ is the hidden representation of comment c and $\vec{mask} = \mathbf{1}\{Ac_{i*} = \vec{c_{nil}}\}, 1 \leq i \leq k$. $\vec{c_{nil}}$ is represented as $\vec{0}$.

The model's architecture uses attention based on *Queries*, *Keys* and *Values* as in [14] but with a single head to determine the importance of each ancestor comment to the main comment. We have decided to use an attention-based mechanism instead of an RNN because a leaf comment is usually invariant to the order of the ancestors. One can refer to any comment from the ancestors but reply to a leaf node.

Note that the weights of the transformer network used to generate embeddings are frozen. The reason for freezing them is that the dataset is minimal, and there is a high chance of overfitting. The loss is calculated using cross entropy $H(y, o)$, as usual. The optimizer used is AdamW [8]. All the hyper-parameters are the default parameters used in Torch Library, except Dropout, in which some experiments need to be higher, varying from 0.6 to 0.85. A CosineAneallingWarmRestart [7]



$$C = \{c_{nil}, c_1, c_2, c_3\}$$

$$T = \{(c_{nil}, c_1), (c_1, c_2), (c_1, c_3)\}$$

$$ancestor_{context}(c_1, 1) = \{c_{nil}\}$$
$$ancestor_{context}(c_2, 1) = \{c_1\}$$
$$ancestor_{context}(c_3, 1) = \{c_1\}$$

Fig. 2. Example of a thread and ancestor contexts, where k = 1

scheduler was also used with the initial reset set to 5 epochs, and the number of epochs required is doubled every reset.

V. Experimental results

To test our model, we randomly shuffled all the Educational rounds, from which we picked 10 for training, 3 for validation and 3 for testing. The Div rounds represent a hidden test. In each training session, we chose the model that yielded the lowest loss on the validation over 50 epochs.

As a baseline model, we choose to apply *TF-IDF* vectorization to the text and a *Random Forest* model. The reason for not selecting a deep learning model as a baseline was that there seemed to be some keywords which could indicate the label. For example, the SolutionExp in most cases contained the words *"Solution"*, *"Approach"* or *"Solve"*. The results can be seen in Table III.

	Educ			Div		
Class	P	R	F1	P	R	F1
Irrelevant	0.70	0.82	0.76	0.88	0.73	0.80
Relevant	0.82	0.70	0.75	0.41	0.66	0.50
Average	0.76	0.76	0.76	0.65	0.69	0.65

TABLE III
TEST RESULTS ON EDUC DATASET WITH TF-IDF AND RANDOM FOREST

For our proposed method, we experimented with the following transformers: BERT [4], RoBERTa [6], and SciBERT [1]. We also used SciBERT as it was trained on scientific papers and thus should provide a better representation for our task. The following scenarios were considered: (1) Using the pretrained transformers for generating embeddings; (2) Finetuning the models on the unlabelled dataset using the MLM task. We apply the subsampling scheme used by Word2Vec [10] to mask the words. Many latex-specific symbols and operators are code-specific, and we want to decrease the chance of masking them. (3) Enhancing the training dataset with weak augmentations by generating six augmented alternatives for each comment. For all experiments the number of ancestors $k = 3$ and the results are presented in Table IV.

In the results from above, it can be seen that Educational rounds are more accessible to predict than Div rounds. As expected, Div rounds contain much more Irrelevant data, which is harder to predict. The results of using just the comments without the ancestors can be found in Table IV.

VI. Comparison against ChatGPT (GPT 3.5)

We decided to compare our results with ChatGPT's results. To have predictions made by ChatGPT, we created an initial message in which we described the task at hand and imposed a format which should be followed in the conversation. All the requests were made using OpenAI's API using gpt-3.5-turbo for the prompt without the thread and gpt-3.5-turbo-16k for the prompt with the thread. The temperature was set to 0.2. We chose a lower temperature to enforce the model to follow the format proposed in the prompt.

The maximum depth of a comment thread is ∼16 and is present in the Div fold. We feed a tree from the thread along the %rule% token in the conversation. We found that if we use the token, the variance of ChatGPT responses is lower. We also tried to specify the correct results at each step, but it didn't bring any benefit. We also tried to offer a more straightforward format for ChatGPT to follow by removing the thread format and providing a comment in which it has to be labelled. The results can be found in Table IV.

VII. Conclusion

In this paper, we proposed the **CFComments** dataset, which contains 19 labelled comment threads and 1121 unlabelled comment threads. We have shown that relevant labels can be extracted from the comments using **CommentThreadFilter**, which can contribute to extending other datasets. We also created an attention-based classifier to tackle the thread structure, obtaining better results than using ChatGPT on the reframed task of Relevant and Irrelevant comments, along with finetuning the initial BERT family models on the unlabelled data and using augmentation methods. A link to the repository, which contains the code and datasets, can be found here **https://github.com/xzzyaa23/CFComments**.

In future work, we would like to use margin-match semi-supervised learning [12] to select the unlabelled observations that should be part of the training in each epoch using the margin metric and pseudo labels. Another approach could be to finetune an LLM like LLama-2 [13] on the unlabelled data

	Comment Tree						Without Comment Tree					
	Educ			Div			Educ			Div		
Method	P	R	F1	P	R	F1	P	R	F1	P	R	F1
BertBaseCased	0.81	0.79	0.79	0.67	0.75	0.63	0.82	0.79	0.79	0.67	0.75	0.63
Roberta	0.82	0.78	0.78	0.66	0.73	0.61	0.82	0.81	0.81	0.67	0.75	0.64
ScibertCased	0.83	0.81	0.81	0.67	0.76	0.64	0.85	0.85	0.85	0.68	0.76	0.67
BertBaseCased + AUG	0.79	0.77	0.77	0.66	0.72	0.60	0.83	0.78	0.78	0.66	0.72	0.60
Roberta + AUG	**0.86**	**0.85**	0.85	0.68	0.76	0.64	0.85	0.84	0.84	0.68	0.76	0.65
SciBertCased + AUG	0.82	0.78	0.78	0.66	0.73	0.60	0.86	0.83	0.83	0.67	0.75	0.63
FinetunedBertBaseCased128	0.85	0.83	0.83	0.68	0.76	0.65	0.85	0.83	0.83	0.68	0.76	0.6
FinetunedBertBaseCased512	0.85	0.85	0.85	**0.70**	0.78	0.70	0.85	0.84	0.84	0.70	0.79	0.70
FinetunedRoberta512	**0.86**	**0.85**	**0.86**	**0.70**	**0.79**	**0.70**	0.85	0.84	0.84	**0.71**	**0.80**	0.70
FinetunedBertBaseCased128 + AUG	**0.86**	**0.85**	0.85	**0.70**	**0.79**	**0.70**	**0.87**	**0.86**	**0.86**	0.70	0.79	0.6
FinetunedBertBaseCased512 + AUG	0.85	**0.85**	0.85	**0.70**	0.78	**0.70**	0.85	0.84	0.84	**0.71**	0.79	**0.71**
FinetunedRoberta512 + AUG	**0.86**	**0.85**	**0.86**	**0.70**	**0.79**	**0.70**	0.85	0.85	0.85	0.70	0.79	0.69
ChatGPT	0.80	0.74	0.74	0.69	0.77	0.63	0.81	0.77	0.77	0.70	0.79	0.68
Tfldf + Random Forest	-	-	-	-	-	-	0.76	0.76	0.76	0.65	0.69	0.65

TABLE IV

TEST RESULTS ON THE LABELLED DATASET USING THE PROPOSED ARCHITECTURE WITH TREE AND WITHOUT. THE MODELS WERE FINETUNED ON THE UNLABELLED FOLD. THE NUMBER AFTER THE MODEL REPRESENTS THE MAXIMUM LENGTH USED. FOR ALL THE CASES WHERE AUGMENTATION IS USED, *+ AUG is added.*

in which we consider each path of the tree as a chat and predict the original labels. We also want to introduce the problem statements and the editorials as part of the comment tree and predict all the labels.

REFERENCES

[1] Iz Beltagy, Arman Cohan, and Kyle Lo. Scibert: Pretrained contextualized embeddings for scientific text. *CoRR*, abs/1903.10676, 2019. URL http://arxiv.org/abs/1903.10676.

[2] Kwangsu Cho. Machine classification of peer comments in physics. In *Educational Data Mining 2008*, 2008.

[3] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20 (1):37–46, 1960.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

[5] KM Kavitha, Asha Shetty, Bryan Abreo, Adline D'Souza, and Akarsha Kondana. Analysis and classification of user comments on youtube videos. *Procedia Computer Science*, 177:593–598, 2020.

[6] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL http://arxiv.org/abs/1907.11692.

[7] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983, 2016. URL http://arxiv.org/abs/1608.03983.

[8] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[9] Amirabbas Majd, Mojtaba Vahidi-Asl, Alireza Khalilian, Ahmad Baraani-Dastjerdi, and Bahman Zamani. Code4bench: A multidimensional benchmark of code-forces data for different program analysis techniques. *Journal of Computer Languages*, 53:38–52, 2019.

[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[11] Andrei Serbanoiu and Traian Rebedea. Relevance-based ranking of video comments on youtube. In *2013 19th international conference on control systems and computer science*, pages 225–231. IEEE, 2013.

[12] Tiberiu Sosea and Cornelia Caragea. Marginmatch: Improving semi-supervised learning with pseudo-margins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15773–15782, 2023.

[13] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

[15] Jing Wang, Clement T Yu, Philip S Yu, Bing Liu, and Weiyi Meng. Diversionary comments under political blog posts. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1789–1793, 2012.