ICIC Express Letters
Part B: Applications
Volume **13**, Number **7**, July **2022**

ICIC International ©2022 ISSN 2185-2766
pp. **749–756**

QUANTITATIVE ARGUMENT SUMMARIZATION USING TEXT-TO-TEXT TRANSFER TRANSFORMER

William Harly, Hansen Riady Kwee and Derwin Suhartono*

Computer Science Department
School of Computer Science
Bina Nusantara University
Jl. K. H. Syahdan No. 9, Kemanggisan, Palmerah, Jakarta 11480, Indonesia
{ william.harly; hansen.kwee }@binus.ac.id; *Corresponding author: dsuhartono@binus.edu

Received August 2021; accepted November 2021

Abstract. *Currently, argument mining and the related fields are expanding, generating numerous amounts of data. Thus, a method to summarize the extracted arguments becomes increasingly important to create a comprehensive summary. In this work, we aim to improve the performance of quantitative summarization where arguments are mapped to a key point defined as a high-level argument. We cast this problem into Recognizing Textual Entailment task and solve it with supervised learning by finetuning Text-to-Text Transfer Transformers. Using this method, we achieved an F1-score of 98.5% using a 4-fold cross-validation method. Compared to previous work, our experiment resulted in a 17.6% increase in performance.*

Keywords: Argument mining, Quantitative summarization, Text-to-Text Transfer Transformer, Recognizing Textual Entailment, Online discussion

1. **Introduction.** The Internet has been used by many people as a platform for discussion due to easy access and the anonymity status gained when accessing it [1]. We can see from Figure 1 that the number of discussions in Reddit, which is one of the popular discussion platforms, keeps increasing each year. With so many discussions happening on the Internet, methods that are able to extract and process arguments become increasingly important. Even now, many studies have been done on this field from extracting argument [2] to classifying argument [3]. However, even though there are many researches on argument mining, there are still very few research efforts that tried to show these arguments in a simple and easy-to-understand summary.

Primarily, the research in summarization is split into two groups: extractive summarization and abstractive summarization. Extractive summarization focuses on finding important sections in the document, while abstractive summarization tries to create a concise summary that might contain phrases that do not exist in the original text [4]. Recent works in extractive summarization consider summarization as a word or sentence level classification problem and addressed it by calculating sentence representations using neural architectures [5,6]. Other work tried to adopt document-level features to rank extractive summaries [7]. On the other hand, in abstractive summarization, several works have aimed to make an abstract summary from texts by making a short representation of a document [8] or compression of the document [9]. Abstractive summarization also has been tried for online political debates [10].

However, these approaches are not as effective when applied in an online discussion because an argument will often be used multiple times. This usage of the same argument multiple times often signifies the participant's understanding of the discussion topic and other information, such as when there is an outlier argument in the discussion that is

DOI: 10.24507/icicelb.13.07.749

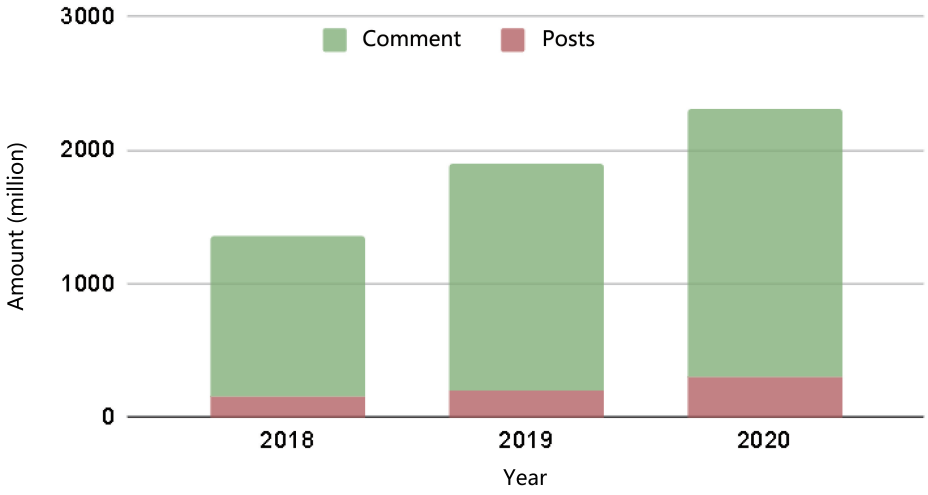

FIGURE 1. Number of posts and comments published on Reddit[1]

thrown around only to aggravate the discussion. However, both abstractive and extractive methods removed this information, which reduces the comprehensiveness of the generated summary. To work around this problem, newer research implemented a clustering algorithm to group similar arguments by their similarity [11] and frames [12]. Unfortunately, these works did not attempt to make a summary out of the created clusters.

Recent work has solved this problem by mapping the argument to a *key point* defined as a high-level argument using several methods, with the best performance acquired using transformer architecture [13]. Although this method has been further improved by swapping the transformer with a more powerful one [14], there is still room for improvement by using a newer and more powerful transformer such as Text-to-Text Transfer Transformer [15].

In both of these previous works [13,14], they used several evaluation policies. For example, when matching the argument to a single keypoint, the remaining suitable key points that are not chosen by the model are regarded as a false negative. Thus, we also try to improve on the evaluation method to better capture the performance of the model.

In summary, this work will improve previous work [13,14] on mapping argument to key points by using a higher-performing transformer, which is Text-to-Text Transfer Transformer [15], and further refining the data preprocessing step. By performing these improvements, we were able to improve the performance on quantitative argument summarization to 98.5% of F1-score.

In Section 2, we examine the works related to our research. The step of our experiment is explained in Section 3. The result of our experiment is shown in Section 4. Finally, our conclusions are presented in Section 5.

2. **Problem Statement and Preliminaries.**

Previous research on summarization can be separated into two categories based on their approach: abstractive summarization and extractive summarization. In abstractive summarization, the generated summaries potentially contain new phrases and sentences that may not appear in the source text [16]. While in extractive summarization, the generated summary is created by taking an important section from the source text [17]. However, quantitative summarization focuses on summarizing discussion by grouping similar arguments and creating a textual representation of each group to create a concise summary. In this section, we examine the previous work in quantitative summarization.

[1]https://redditblog.com/2020/12/08/reddits-2020-year-in-review/; https://redditblog.com/2019/12/04/reddits-2019-year-in-review/; https://redditblog.com/2018/12/04/reddit-year-in-review-2018/

One such work proposed a method for automatic multi-document summarization which consisted of two clustering stages [18]. In the first stage, the clustering is performed on the document level based on topic similarity. For the second stage, the clustering is performed on the sentence level based on semantic and syntactic similarity. Each cluster is then evaluated using a summary builder checker to find the best summary based on the coherence and readability of the generated summary.

Another work that is more focused on summarizing arguments used a similarity-based approach to decide whether two arguments are under the same topic [11]. They used argument pairs from an argument search engine containing a lot of noise arguments unrelated to the target topic and train a supervised similarity function based on a transformer, which significantly improved the evaluation result compared with unsupervised methods on the same arguments pair. This work concludes that fine-grained semantic nuances used by the supervised method determine the similarity between arguments and not the lexical used in the unsupervised method.

There is also another study on clustering arguments by grouping multiple arguments in a topic into several non-overlapping *frames* [12]. Frame refers to a group of arguments that focus on the same aspect. From this frame, they clustered the argument using two different methods, first by removing an argument's topic features, and the second uses conclusion and premise. The result of this work shows the benefit of removing topic features and using argument conclusions to identify the argument's frame.

However, using only the clustering method to create quantitative summarization means that we still need to create a textual representation from the clustering results. Thus, a recent work introduces a new summarization method, where each cluster should represent a pre-selected *key point* [13]. This research tried several approaches to solve this new task, with the best performing model using supervised learning with pre-trained transformer model BERT.

This work is further improved by using several other pre-trained transformer models with a better performance compared to BERT, especially on Recognizing Textual Entailment (RTE) [14]. The most notable result is obtained by using ALBERT [19] achieving an F1 of 78.4%. Although ALBERT is able to bring a significant performance improvement, there is still room for improvement by using a newer transformer model. In the same research [14], they also explore the notion of automatic key point extraction by using arguments to keypoint matching model to choose a key point with the highest level of coverage.

Both of these works [13,14] use several evaluation policies to better suit the dataset used due to how most arguments have at most one matched key point. For example, when matching the argument to a single keypoint, the model will choose the best matching key point regardless of the confidence score. However, this method can be further refined by a more thorough preprocessing by removing excess keypoint from the argument with multiple matching keypoint to better capture the model's performance.

Text-to-Text Transfer Transformer. In our experiment, we use transformer, a common approach in achieving state-of-the-art performance on various NLP tasks. The transformer architecture consists of multiple encoder-decoder architectures that use self-attention, a mechanism related to the different positions of a single sequence to create a contextual embedding. This method was able to reduce sequential operation needed and increase the number of processes that can be parallelized. Combined with transfer learning, transformer architecture is able to achieve state-of-the-art performance [20].

The specific transformer used in this experiment is a Text-to-Text Transfer Transformer (T5) [15]. We focused on experimenting with T5 because it is made by scaling up various insights from previous transformers with state-of-the-art performance, thus achieving an even better result than other transformers on various tasks, such as Recognizing Textual Entailment and Semantic Textual Similarity. To ease the model evaluation on these various

NLP tasks, the authors of this work cast all tasks into a "text-to-text" format, with text as both input and output, and different prefixes on the input text for each task.

3. **Methodology.** Based on previous work in previous sections, we performed our research according to the process shown in Figure 2. The research step is similar to prior works on quantitative argument summarization [13,14] with a more thorough preprocessing at the dataset preparation and different hyperparameters at the training phase.

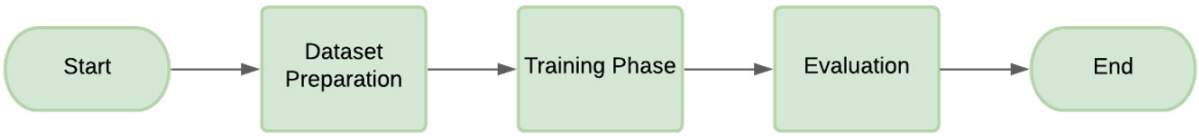

FIGURE 2. Step of argument summarization research

3.1. **Dataset preparation.** The dataset used was the IBM Debater(R)-ArgKP dataset [13], which is a subset of the IBM ArgQ-Rank-30kArgs dataset [21]. The ArgKP dataset contains 24,093 argument and keypoint pairs, which is composed of 6,515 unique arguments related to 28 controversial topics. From these 6,515 unique arguments, not all of them are matched to exactly one key point (having more than one key point or no matching key point). The dataset consists of five fields depicted in Table 1.

TABLE 1. Fields of each instance in the dataset

Label	Description
topic	The title of the debate with the stance
argument	A pro or con argument related to the topic
key_point	A candidate key point
label	1 if a keypoint represents the argument, 0 otherwise

This dataset is preprocessed by removing unused fields such as topic and stance. Afterward, unnecessary symbols are also removed, and all letters are changed into lowercase. The preprocessed dataset is split accordingly to perform 4-fold cross-validation, each fold consists of 7 test topics, 4 development topics, and 17 train topics. Finally, several versions of the test datasets are made by filtering the dataset to match the evaluation policy explained in the evaluation section as follows:

a) Multiple keypoint dataset, a version where the dataset contains arguments with one, multiple, or no matching key points;

b) Single keypoint dataset, a version where the dataset contains only arguments with one matching keypoint. This dataset is made by removing arguments with no matching key points and removing excess matching key points from the test dataset;

c) One or no keypoint dataset, a version where the dataset contains arguments with one or no matching keypoint. This dataset is made by removing excess matching keypoint from the test dataset.

3.2. **Training phase.** All the training is done in the cloud using Tensor Processing Unit (TPUv2-8) and Cloud Storage provided by Google. We used the T5 library[2] that is implemented using TensorFlow mesh [22], allowing for TPU utilization to shorten training time. The text-to-text models used for this experiment are T5-base (220M parameter), T5-large (770M parameter), and T5-3B checkpoint[3] (2.8B parameter) after pre-trained for

[2]https://github.com/google-research/text-to-text-transfertransformer

[3]gs://t5-data/pretrained_models

1 million steps. Compared to previous work [14] which uses BERT-large (336M parameter), RoBERTa-large (355M parameter), and ALBERT-xxlarge-v1 (223M parameter), the number of parameters of the text-to-text model does not differ significantly, especially on the base version of the text-to-text model.

We then used the provided T5 checkpoints without any modification. We convert our task to T5 RTE task format with a maximum input sequence length of 128. The model is fine-tuned using AdaFactor optimizer [23] and 0.1 probability for each dropout layer in the model [15]. The fine-tuning is performed for 27,000 steps with a batch size of 16 and a constant learning rate of 3e-4 when the sign of overfitting started to appear on validation data. A few examples of the input-output in T5 RTE format are shown in Figure 3.

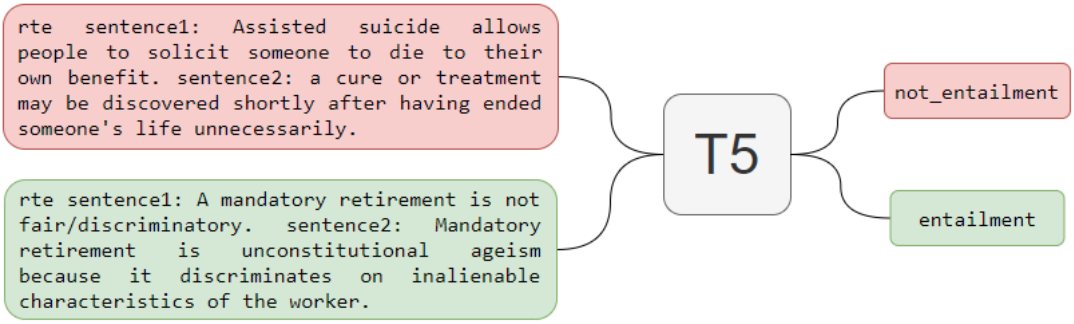

FIGURE 3. A few examples of the input-output in T5 RTE format

3.3. **Evaluation.** After the training phase, we run each model on the test dataset to perform a prediction on the given argument and key point pair. Alongside the prediction, we also measure the confidence score of the model in the form of log-likelihood using the provided utilities in the T5 library. And then, we perform the following step to acquire the model performance.

Step 1: Using the *Loglikelihood* of each prediction, we convert it back into probability (P) using the equation

$$P = e^{Loglikelihood} \tag{1}$$

Step 2: Using the probability (P) and the original model prediction output (y), we calculate the match score between an argument with a candidate keypoint using Equation (2).

$$f(P, y) = \begin{cases} 0.5 + \frac{1}{2}P, & y = entailment \\ 0.5 - \frac{1}{2}P, & y = not\ entailment \end{cases} \tag{2}$$

Step 3: We then implemented a moving threshold [24] by selecting the threshold t that maximizes the F1-score on the positive class over the development set using the resulting match score. This is done to negate the imbalance proportion of the class in the training dataset.

Step 4: From the resulting match score, the final prediction of the model is made based on the following selection policies used in previous work [13].

a) *Threshold* (TH) policy matches each argument in the test data to multiple key points with a match score above a certain threshold t. This policy may lead to an argument with no matching key point.

b) *Best Match* (BM) policy groups the test data by argument and matches each argument with one key point with the highest match score. This policy is best used when each argument must have one matching key point.

c) *Best Match + Threshold* (BM+TH) policy groups the test data by argument and matches each argument to one key point with the highest match score and above a certain threshold t. This policy is best used when each argument has either one or no matching key point.

Using the final prediction of the models, we evaluate each model by measuring accuracy, precision, recall, and F1 [25] of all folds on the positive class, before averaging them to get the final model performance.

4. **Main Results.** Table 2 shows our result on matching the argument to key points with several policies using a Text-to-Text Transfer Transformer compared to the best method from previous works [13,14]. All of the results are acquired by averaging the performance matrix across the different folds.

TABLE 2. Argument to key point matching results, across different evaluation policies

Model	Selection policy	Accuracy	Precision	Recall	F1
BERT-large [13]	TH	0.867	0.677	0.700	0.685
	BM	0.879	0.705	0.716	0.710
	BM+TH	0.893	0.788	0.665	0.721
ALBERT-xxlarge-v1 [14]	TH	0.909	0.779	0.794	0.784
	BM	0.908	0.778	0.785	0.780
	BM+TH	0.926	0.877	0.751	0.809
RoBERTa-large [14]	TH	0.897	0.731	0.803	0.765
	BM	0.895	0.745	0.753	0.749
	BM+TH	0.913	0.849	0.711	0.773
T5-base	TH	0.944	0.789	0.999	0.881
	BM	0.980	0.960	0.960	0.960
	BM+TH	0.974	0.914	0.959	0.935
T5-large	TH	0.949	0.856	0.932	0.881
	BM	0.981	0.962	0.962	0.962
	BM+TH	0.971	0.946	0.907	0.921
T5-3B	TH	0.963	0.877	0.964	0.914
	BM	0.992	0.985	0.985	0.985
	BM+TH	0.983	0.966	0.952	0.957

Using the text-to-text model, we were able to outperform previous work across all the performance metrics. The smallest T5 model (T5-base) used in this experiment has a similar amount of parameters with ALBERT-xxlarge-v1, but is still able to improve performance on all metrics. We also observe that the performance increase with the number of parameters in the T5 model.

These performance increases are in line with our conjecture since T5 is a better model not only because it is a scaled-up model but also built based on insights from other current state-of-the-art transformers. The use of the predefined format for T5 RTE task also eases the downstream process of Recognizing Textual Entailment task to mapping argument with key points.

Our method is able to achieve significantly higher performance on selecting a single matching key point (BM) even with the same model size compared to previous work [14]. This is due to our preprocess where we remove arguments with no matching key point and removing excess matching key points from test data. Additionally, this preprocessing also resulted in the value of precision, recall, and F1 to have the same value on the single matching key point due to the nature of force matching of argument to a single key point, resulting in the same amount of false positive and false negative.

5. **Conclusion.** Based on the result of the experiment, it can be concluded that we can achieve higher performance in quantitative summarization by mapping arguments to key points using Text-to-Text Transfer Transformer (T5), which is a scaled-up model built based on insights from other current state-of-the-art transformers. We were able to increase the performance of the task compared to previous work with 13.0% on multiple key points, 20.5% on a single key point, and 14.8% on a single or no key point measured by F1.

Although the text-to-text model has yielded a high performance, there is still room for improvement and further research. In this research paper, we only experimented using T5-3B with 3 billion parameters, while the original paper reports a higher performance on the T5-11B with 11 billion parameters. Future work also can experiment on using our method for automatic keypoint extraction to further improve the automation of the argument summarization process.

REFERENCES

[1] E. M. Onyema, E. C. Deborah, A. O. Alsayed, N. N. Quadri and S. Sanobe, Online discussion forum as a tool for interactive learning and communication, *International Journal of Recent Technology and Engineering (IJRTE)*, vol.8, no.4, pp.4852-4859, 2019.

[2] I. Persing and V. Ng, Unsupervised argumentation mining in student essays, *Proc. of the 12th Conference on Language Resources and Evaluation (LREC2020)*, pp.6795-6803, 2020.

[3] R. Winata, E. G. Haryono and D. Suhartono, Towards better argument component classification in English essays, *ICIC Express Letters, Part B: Applications*, vol.12, no.2, pp.111-119, 2021.

[4] D. Huang, L. Cui, S. Yang, G. Bao, K. Wang, J. Xie and Y. Zhang, What have we achieved on text summarization?, *Proc. of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp.446-469, 2020.

[5] R. Nallapati, F. Zhai and B. Zhou, SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents, *Proc. of the 31st AAAI Conference on Artificial Intelligence*, pp.3075-3081, 2017.

[6] J. Xu and G. Durrett, Neural extractive text summarization with syntactic compression, *Proc. of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp.3292-3303, 2019.

[7] M. Zhong, P. Liu, Y. Chen, D. Wang, X. Qiu and X. Huang, Extractive summarization as text matching, *Proc. of the 58th Annual Meeting of the Association for Computational Linguistics*, pp.6197-6208, 2020.

[8] L. Wang and W. Ling, Neural network-based abstract generation for opinions and arguments, *Proc. of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp.47-57, 2016.

[9] J. Zhou and A. Rush, Simple unsupervised summarization by contextual matching, *Proc. of the 57th Annual Meeting of the Association for Computational Linguistics*, pp.5101-5106, 2019.

[10] C. Egan, A. Siddharthan and A. Wyner, Summarising the points made in online political debates, *Proc. of the 3rd Workshop on Argument Mining (ArgMining2016)*, pp.134-143, 2016.

[11] N. Reimers, B. Schiller, T. Beck, J. Daxenberger, C. Stab and I. Gurevych, Classification and clustering of arguments with contextualized word embeddings, *Proc. of the 57th Annual Meeting of the Association for Computational Linguistics*, pp.567-578, 2019.

[12] Y. Ajjour, M. Alshomary, H. Wachsmuth and B. Stein, Modeling frames in argumentation, *Proc. of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp.2922-2932, 2019.

[13] R. Bar-Haim, L. Eden, R. Friedman, Y. Kantor, D. Lahav and N. Slonim, From arguments to key points: Towards automatic, *Proc. of the 58th Annual Meeting of the Association for Computational Linguistics*, pp.4029-4039, 2020.

[14] R. Bar-Haim, Y. Kantor, L. Eden, R. Friedman, D. Lahav and N. Slonim, Quantitative argument summarization and beyond: Cross-domain key point analysis, *Proc. of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp.39-49, 2020.

[15] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li and P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, *Journal of Machine Learning Research*, vol.21, pp.1-67, 2020.

[16] L. Liu, Y. Lu, M. Yang and Q. Qu, Generative adversarial network for abstractive text summarization, *The 32nd AAAI Conference on Artificial Intelligence (AAAI-18)*, pp.8109-8110, 2018.
[17] N. Moratanch and S. Chitrakala, A survey on extractive text summarization, *2017 International Conference on Computer, Communication and Signal Processing (ICCCSP)*, pp.1-6, 2017.
[18] K. A. Maria, K. M. Jaber and M. N. Ibrahim, A new model for Arabic multi-document text summarization, *International Journal of Innovative Computing, Information and Control*, vol.14, no.4, pp.1443-1452, 2018.
[19] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma and R. Soricut, ALBERT: A lite BERT for self-supervised learning of language representations, *The 8th International Conference on Learning Representations (ICLR2020)*, 2020.
[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, A. N. Gomez, L. Kaiser and I. Polosukhin, Attention is all you need, *The 31st Conference on Neural Information Processing Systems (NIPS2017)*, 2017.
[21] S. Gretz, R. Friedman, E. Cohen-Karlik, A. Toledo, D. Lahav, R. Aharonov and N. Slonim, A large-scale dataset for argument quality ranking: Construction and analysis, *Proc. of the AAAI Conference on Artificial Intelligence*, vol.34, no.5, pp.7805-7813, 2020.
[22] N. Shazeer, Y. Cheng, N. Parmar, D. Tran, A. Vaswani, P. Koanantakool, P. Hawkins, H. Lee, M. Hong, C. Young, R. Sepassi and B. Hechtman, Mesh-TensorFlow: Deep learning for supercomputers, *arXiv.org*, arXiv: 1811.02084, 2018.
[23] N. Shazeer and M. Stern, AdaFactor: Adaptive learning rates with sublinear memory cost, *CoRR*, 2018.
[24] F. Provost, Machine learning from imbalanced data sets 101, *The AAAI 2000 Workshop on Imbalanced Data*, 2000.
[25] D. Powers, Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation, *Mach. Learn. Technol.*, 2008.