Arguments to Key Points Mapping with Prompt-based Learning

Ahnaf Mozib Samin **Behrooz Nikandish** **Jingyan Chen**

University of Groningen
Groningen, The Netherlands
{asamin9796, behrooz.nikandish, chenjingyan0722}@gmail.com

Abstract

Handling and digesting a huge amount of information in an efficient manner has been a long-term demand in modern society. Some solutions to map key points (short textual summaries capturing essential information and filtering redundancies) to a large number of arguments/opinions have been provided recently (Bar-Haim et al., 2020). To complement the full picture of the argument-to-keypoint mapping task, we mainly propose two approaches in this paper. The first approach is to incorporate prompt engineering for fine-tuning the pre-trained language models (PLMs). The second approach utilizes prompt-based learning in PLMs to generate intermediary texts, which are then combined with the original argument-keypoint pairs and fed as inputs to a classifier, thereby mapping them. Furthermore, we extend the experiments to cross/in-domain to conduct an in-depth analysis. In our evaluation, we find that i) using prompt engineering in a more direct way (Approach 1) can yield promising results and improve the performance; ii) Approach 2 performs considerably worse than Approach 1 due to the negation issue of the PLM.

1 Introduction

With internet technology getting more accessible to the general public, a flood of information in the digital space can be observed. On online social media, people tend to provide arguments/counter-arguments on various topics, including government policies, movie reviews, and controversial issues such as gun control, abortion, and global climate changes, etc. This kind of information is valuable for government policymakers, business people, and academicians who conduct research on societal changes over time. However, due to the abundance of arguments, it becomes nearly impossible to go through each one manually and make a decision. Moreover, manually reading the arguments does not allow systematic categorization, making it unlikely to quantify them.

To address the issue, (Bar-Haim et al., 2020) first proposed a method to categorize the arguments by mapping them to a set of pre-defined key points set by the domain experts. They fine-tuned a pre-trained language model (PLM) using the ArgKP dataset they built. Fine-tuning PLMs has been proved to achieve superior results over the conventional approach of training a neural network model from scratch. However, there are several limitations to directly fine-tuning PLMs. First, fine-tuning a PLM requires a substantial amount of data and computational resources for each downstream task. Second, the typical way of directly fine-tuning the PLMs does not simulate how the human brain performs NLP tasks. Humans need to be prompted by providing additional task-specific information at first. For example, if we want to know whether a review is positive, negative, or neutral from a human, we would prepare a question like "Do you think the review is positive, negative, or neutral?" to prompt the human to accomplish the task.

Prompt-based learning, built on language models that model the probability of text directly, has been a recent revival in NLP and has shown great potential to address the above limitations. (Brown et al., 2020) indicated that developing a very large PLM with 175 billion tokens and prompting the PLM alleviates the need for additional data for fine-tuning. Thus, it allows us to perform zero-shot and few-shot learning for several NLP tasks. Motivated by this, we exploit prompt-based learning to accomplish the argument-to-keypoint summarization task. More precisely, we would like to shed light on the following research questions:

- Does prompt-based learning allow better utilization of the PLMs for the argument-to-keypoint mapping task? In other words, can it outperform the typical direct fine-tuning PLMs approach?

- What are the challenges that arise with imple-

menting prompt-based learning for this task?

Our contributions are mainly two-fold:

- First, we implement prompt-based learning for the argument-to-keypoint mapping task for the first time, to the best of our knowledge, and compare the results with conventional fine-tuning approaches. To this end, we fine-tune the T5-base PLM with five different prompt templates and report their final F1-scores.
- Second, we propose a novel architecture that takes an argument as input using prompt-based learning and generates an intermediary text after fine-tuning the PLMs. Then, we employ several machine learning classifiers to decide whether the argument, key point, and the intermediary text triple are a match. We demonstrate and analyse the promising results and shortcomings of the proposed architecture.

2 Related Work

Some researchers have done well-executed and rigorous studies and provided thoughtful methods in the field of argument-to-keypoint summarization. (Bar-Haim et al., 2020) established the ArgKP dataset which is the first large-scale dataset for this task and proposed a method to automatically map many arguments to a small number of given key points. They analysed and evaluated some unsupervised methods with TF-IDF and word embeddings and supervised methods like fine-tuning Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018). This study made an excellent basis for next research in this field and is also the foundation of our project. To improve the performance of this task, (Kapadnis et al., 2021) leveraged existing state-of-the-art PLMs along with incorporating additional datasets (IBM Rank 30k and STS) and features like the topic of arguments. But the main shortcoming of these two studies is that the key points are pre-defined by expert annotators, which is an obstacle to making the process fully automatic.

Later, (Bar-Haim et al., 2020) made a more in-depth study to promote the previous line of research, and developed a method for extracting key points automatically from a set of comments, which allows fully automatic key point analysis. And they compared more PLMs including BERT-large-uncased (Devlin et al., 2018), XLNett-large-cased (Yang et al., 2019), RoBERTa-large (Liu et al., 2019), and ALBERT-xxlarge-v1 (Lan et al., 2020), in terms of run time and accuracy, which showed a significant improvement above their previous best results in (Bar-Haim et al., 2020). However, during the step of the automatic key point extraction process, they considered only single sentences and filtered out long sentences as well as those sentences that start with pronouns. Consequently, the model likely misses some potential key points.

Prompt engineering has recently become an emerging field of study in NLP. (Liu et al., 2021) introduced the basics of this new paradigm in detail, and (Brown et al., 2020) confirmed the advantages of adopting prompt-based learning on various NLP tasks such as question answering, translation, and probing tasks for common sense reasoning. And prompt engineering techniques also work well on probing factual knowledge in language models (Jiang et al., 2020). Nonetheless, the suitability of using prompt-based learning for a wide variety of NLP tasks has yet to be proven, and prompt-based learning has not been explored deeply in argument to key point summarization. In this work, we employ this promising paradigm to improve the task of argument-to-keypoint mapping further and provide two approaches to examine the performance of prompt-based learning.

3 Methods

We explore two approaches for this task and compare them with our baselines without any prompt engineering technique. Approach 1 aims to make classification using prompt-based learning. And Approach 2 consists of text generation and text classification with the help of prompt engineering. We use three different Transformer-based (Vaswani et al., 2017) PLMs (BERT (Devlin et al., 2018), BART (Lewis et al., 2020) and T5 (Raffel et al., 2020)) to implement these approaches. The following subsections describe how these PLMs work and why they are appropriate for this task. Moreover, we introduce prompt engineering and the structure of the two approaches in this section.

3.1 Pre-trained Language Models

BERT Unidirectional pre-train architectures limit the choice of architectures during pre-training.

For instance, utilizing left-to-right architecture like in OpenAI GPT (Radford et al., 2018), each token can only attend to previous tokens in the self-attention layer of the Transformer. (Devlin et al., 2018) proposed BERT to alleviate the limitations of unidirectional architectures using *a masked language model*. The architecture of the model is a multi-layer bidirectional Transformer encoder. The model is pre-trained utilizing two unsupervised tasks: Masked Language Models and Next Sentence Prediction (NSP). In many downstream tasks as well as the argument-to-keypoint task, understanding the relationship between two sentences is critical. The BERT model is pre-trained for a binarized NSP task to train the model to understand sentence relationships, which makes the BERT model a good choice for the key point analysis task.

BART BART (Lewis et al., 2020) is a PLM that combines Bidirectional and Auto-Regressive Transformers. The denoising autoencoder is built using a sequence-to-sequence model and it can be applied to various downstream tasks. It uses a standard Transformer-based neural machine translation architecture with a bidirectional encoder and a left-to-right decoder. During the pre-training process, an arbitrary noise function is applied to the input text, and then a sequence-to-sequence model is responsible for reconstructing the original text. Section 3.3.2 describes Approach 2 in which our model generates an intermediary text. BART can be a reasonable choice for this task because it performs effectively in text generation (Yuan et al., 2021) and text summarization (Huang et al., 2020) tasks.

T5 (Raffel et al., 2020) proposed a unified text-to-text Transformer-based model to explore the limitations of transfer learning using an encoder-decoder architecture. It comprises an encoder that maps the input words from the source language to an output representation. The decoder is a conditional language model that attends to the encoder representation and generates target words one by one, based on the source word and previously generated target language words at each time step. The main idea behind this model is to consider all text processing tasks as a text-to-text problem, feeding the model a text as input and generating new text as output. This provides the ability to apply the model, loss function, hyperparameters, and other parameters to various tasks, including machine translation, text summarization and classification, and question answering. We plan to use T5 in both Approach 1 and 2.

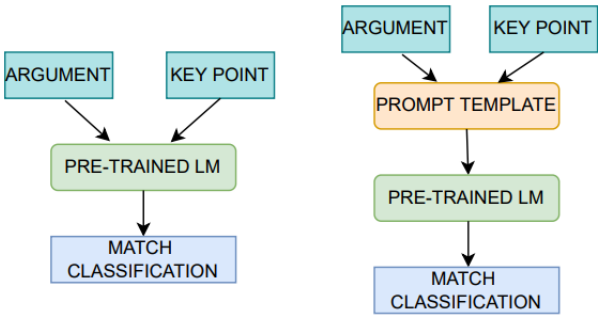

Figure 1: The architecture of baseline (left) and Approach 1 (right)

3.2 Baseline

The architecture of our baseline which is shown in Figure 1 on the left is similar to (Bar-Haim et al., 2020)'s work. We build a classifier to identify whether a pair of *(argument, key point)* is matched or not. To this end, we fine-tune four PLMs, BERT-base, BERT-large, T5-base, and T5-small. We train the models with the train set first and adjust the parameters like epoch with the dev set, and the trained model is evaluated using the unseen data from the test set finally. We do not employ any prompt engineering in the baselines to compare our results with other prompt-based learning approaches in this work.

3.3 Prompt Engineering

To train a model in traditional supervised learning, it is required to have large amounts of supervised data for the task at hand. Prompt-based learning approaches are an attempt to get around this problem. We will first explain the basic form of prompting, and then show how we adopt prompting techniques in the argument-to-keypoint mapping task.
(Liu et al., 2021) described the basic prompting process in three steps: The first step is *prompt addition*, in which a *prompting function* is defined to pre-process the input text. This step consists of two processes:

1. Creating a *template*, which consists of some fixed extra tokens and two slots: *input slot* [X] for input text and *answer slot* [Z] for predicted output that will be used in the *answer mapping* step.

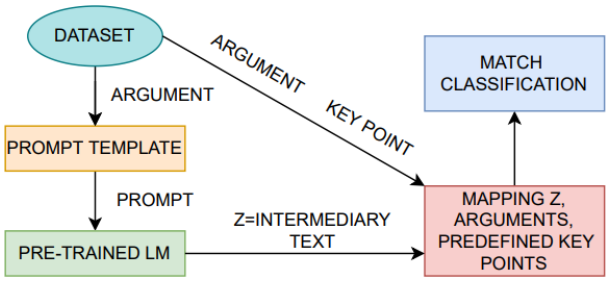

Figure 2: The architecture of Approach 2

2. Filling input slot [X] with the input text.

The output slot [Z] could be either in the middle of the template (*cloze prompt*) or at the end (*prefix prompt*). Depending on the task, the number of input and output slots can vary freely. The second step is *answer search*. In this step, the output slot [Z] in the prompt will be filled by a potential answer, which is the highest scoring answer. In the last step, *answer mapping*, the highest-scoring answer will be mapped to the highest-scoring output. This is the case in text generation tasks, but in some tasks like text classification, each potential answer has a corresponding output to be mapped to.
To answer the research question, we use prompt engineering techniques in two separate approaches for cross/in-domain to investigate if prompt-based learning can outperform our baselines. The architectures of our approaches are discussed in the following subsections.

3.3.1 Approach 1

In Approach 1, we use prompt-based learning in conjunction with fine-tuning a PLM. As the architecture illustrated in Figure 1 on the right, the *(argument, key point)* pairs are transformed to the given prompt template and fed to the T5-base as input. We try various templates shown in table 2 to see how different templates influence our results. In these templates, taking *(argument, key point)* as input texts, [X1] and [X2] represent the *argument* and the *key point* respectively. The answer space of [Z], which is the output text, can be either *matched/not matched* or *Yes/No*, depending on the chosen template. The following example shows the process of creating an input text using a prompt template:

- **Argument [X1]**: *Urbanization destroys the enviroment, and mankind should be finding ways of utilising the space already occupied more efficiently instead*
- **Key point [X2]**: *Urbanization harms the environment*
- **Answer space [Z]**: *matched, not matched*
- **Template**: *The argument: [X1] is [Z] with the key point: [X2]*
- **Input text**: *The argument: urbanization destroys the enviroment, and mankind should be finding ways of utilising the space already occupied more efficiently instead, is* ***matched*** *with the key point: Urbanization harms the environment.*

3.3.2 Approach 2

For Approach 2, we want to explore the effect of adding additional context based on the prior knowledge of the PLMs. Figure 2 illustrates the architecture of Approach 2. First, the input argument is transformed into the given prompt template as [X] and then is fed to the trained PLM (T5-small or BART-large), which is used for text summarization. The output slot [Z] is filled by a generated summary that is called as *intermediary text* in this paper. To note that we use different templates displayed in Table 4 for matching/non-matching pairs to generate the corresponding intermediary texts, and the templates for two types of pairs have totally opposite connotations (e.g., `mean-not mean` and `correct-wrong`). Lastly, different classifiers are built to determine whether the generated intermediary text, argument, and keypoint triple is matching or not. In this step, we fine-tune BERT-base and T5-small PLMs and apply three machine learning algorithms (Naive Bayes (McCallum et al., 1998), Support Vector Machine (SVM) (Cortes and Vapnik, 1995), Decision Tree(Quinlan, 1986)) with TF-IDF features.

4 Experiments

This section will introduce the ArgKP dataset we use and elaborate on how we processed the data and carried out the experiments.

4.1 Data

We use the established ArgKP dataset (Bar-Haim et al., 2020) in this project. The arguments in ArgKP revolve around 28 disputed topics, and they are a subset of the IBM-Rank-30k dataset (Gretz et al., 2020). The key points were authored by an expert on those topics. Crowd annotations

Topic	Argument	Key point	Stance	Label
We should abandon the use of school uniform	we should not abandon the use of school uniforms because it allows children to not be concerned with competitiveness while attempting to learn.	Children can still express themselves using other means	-1	0
We should adopt atheism	we should adopt atheism because religion causes too much tension and disagreements.	Atheism should be adopted since we cannot prove that God exists	1	0
We should end mandatory retirement	mandatory retirement is a good way of refreshing the workforce, motivating those lower in the pecking order and creating employment opportunities.	A mandatory retirement age creates opportunities for other workers	-1	1

Table 1: Examples from the ArgKP dataset. **Stance** means the argument support(1) or oppose(-1) the topic; **Label** represents the key point is matching (1) or non-matching (0) with the argument.

Experiment	Class	Number of samples per set						Total samples	
		Train		Dev		Test			
		Count	(%)	Count	(%)	Count	(%)	Count	(%)
Cross-domain	All	17,019	70.6	2,903	12.1	4,171	17.3	24,093	100.0
	Matching	3,510	14.5	728	3.0	760	3.1	4,998	20.7
	Non-matching	13,509	56.1	2,175	9.1	3,411	14.2	19,095	79.3
In-domain	All	17,021	70.6	2,904	12.1	4,168	17.3	24,093	100.0
	Matching	3563	14.8	593	2.5	842	3.5	4998	20.7
	Non-matching	13458	55.8	2311	9.6	3326	13.8	19,095	79.3

Table 2: Dataset distribution for cross/in-domain experiments

were gathered to see if a keypoint represented or matched an argument, which resulted in *(argument, key point)* pairs. As shown in the Table 1, each pair is assigned a matching or non-matching label and a stance towards the topic.

There are 24,093 labeled argument-keypoint pairs, and 20% of them are matching/positive pairs. Table 2 displays the distribution of each data split set for cross/in-domain experiments. For the cross-domain experiments, we split the whole dataset according to the number of topics, and each topic only occurs once. We assign 19 topics to the train set, and the dev and test sets contain 4 and 5 topics, respectively. The argument-keypoint pair ratio of the three sets is 71:12:17. For the in-domain experiments, we use the same pair ratio of three split sets as the cross-domain experiments, and each split set includes all of those 28 topics.

4.2 Pre-processing

The ArgKP dataset is well-structured and clean enough so that we do not do much pre-processing except for some basic steps. Some arguments and all key points in the dataset do not contain full stops at the end of the sentence, so our first step is to add full stops for each full sentence if they are missing. The second step is tokenization. The PLMs (BERT, BART, T5) we mainly utilize expect a sequence of tokens as an input, so the tokenizers those PLMs were trained on are employed to tokenize the texts. For the machine learning algorithms (SVM, Naive Bayes, Decision Tree), we tokenize the texts and remove stop words using NLTK python package (Bird et al., 2009).

4.3 Experiment Setup

We implement Approach 1 using OpenPrompt (Ding et al., 2021) framework[1], which is an extensible and open-source toolkit for prompt engineering. We replicate their code to train T5-base using the ArgKP dataset for this task. The code associated with this paper is available on a GitHub repository.[2]

Table 3 contains the some of the hyperparameters of each PLM that is fine-tuned in the baselines and Approach 1 and 2.

PLM	Learning Rate	Epoch	Optimizer
BERT-base BERT-large	2e-5	3	Adam
T5-base	1e-3/1e-4	3	Adam
T5-small	3e-4	4	Adam
BART-large	2e-5	5	Adam

Table 3: Hyperparameters used for finetuning different PLMs

4.4 Evaluation

Only about 20% pairs in the dataset are matching/positive pairs, which means the class distribution is quite imbalanced, and standard metrics such as classification accuracy would be misleading in our case. Therefore, we adopt the macro-averaged F1-score, which takes the arithmetic mean of all the per-class F1-scores as the evaluation method.

In addition, we also attempt threshold metrics in order to handle the imbalance problem. Thresholds are learned from the dev set by maximizing the macro-averaged F1-score. Pairs whose matching score exceeds the learned threshold are considered matched. However, we think it is unfair to compare the results of Approach 1/2 and the baselines with different thresholds. Furthermore, most of the learned thresholds are 0.5, which is the same as the default threshold of binary classification. Accounting for these reasons, we ignore threshold metrics finally.

5 Results & Discussion

5.1 Comparison between baseline and the two approaches

Table 4 shows the comparison between our baselines and the two approaches for both in-domain and cross-domain experiments. We have four baselines that do not incorporate prompt engineering. BERT-base outperforms the rest of the four models, getting an F1-score of 88.4% in the in-domain experiment and 72.0% in the cross-domain experiment. For Approach 1, which utilizes prompt engineering and fine-tuning T5-base with the templates, we get higher F1-scores for each of the five prompt templates examined in this study compared to our baselines. Using the five templates, we achieve almost similar F1-scores for the in-domain experiments, while variations in the F1-scores can be observed for the cross-domain evaluation. T1 template obtains the highest F1-scores with 91.4% for the in-domain and 76.1% for the cross-domain experiments. T2 also achieves the second-best F1-score of 91.0% and the equal F1-score to T1. However, T3 and T4 (template with a definition of the key point) can get F1-scores below 74%.

As mentioned in section 3.3.2, our Approach 2 explores T5-small and BART-large by fine-tuning them with two prompt templates (T6 and T7) to get the intermediary texts. Then a classifier decides whether this is a match or non-match based on the argument, intermediary text, and key point as inputs. In the case of the T6 template, with fine-tuning the BART-large for getting the intermediary texts and using BERT-base as a classifier, we achieve the highest F1-scores of 89.2% and 71.2% for in-domain and cross-domain experiments, respectively. But the best-performing system using the T7 template utilizes T5-small to get the intermediary texts and BERT-base and T5-small as classifiers for in-domain and cross-domain experiments, respectively. The final F1-scores from the best-performing model using the T7 template are 90.0% and 69.8% for in-domain and cross-domain experiments, accordingly. This experiment shows that the T6 template is more suitable for BART-large, whereas the T7 template works well with T5-small. The F1-scores using the Naive Bayes, SVM, and Decision Tree as classifiers are poor compared to T5-small and BERT-base.

Comparing the best-performing models of the baselines, Approach 1 and Approach 2, it is evident that Approach 1 outperforms the baseline for both in-domain and cross-domain datasets. Approach 1 also gets substantial improvement in F1-score getting 76.1%, compared to Approach 2, which gets 69.8% for the cross-domain experiment. While the difference in F1-scores between Approach 1 and 2

[1] https://github.com/thunlp/OpenPrompt

[2] https://github.com/samin9796/arg2keypoint

	Prompt Template	PLM for Intermediary Text	Model	F1-score in-domain	F1-score cross-domain
Baseline	-	-	T5-small	0.866	0.700
			T5-base	0.842	0.682
			BERT-base	0.884	0.720
			BERT-large	0.880	0.709
Approach 1	T1: The argument: [X1] and the keypoint [X2] are [Z].	-	T5-base	**0.914**	**0.761**
	T2: The argument: [X1] is [Z] with the keypoint: [X2]			0.910	**0.761**
	T3: Does the argument: [X1] comprise the fact that [X2]? [Z]			0.908	0.732
	T4: A keypoint is a summarization of the corresponding argument. In other words, an argument comprises a keypoint. Does the argument: [X1], comprise the keypoint [X2]? [Z]			0.913	0.737
	T5: Argument: [X1] Keypoint: [X2] $"soft" : "Does"$ $"soft" : "the", "soft_id" : 1$ argument matches $"soft_id" : 1$ keypoint? [Z]			0.911	0.754
Approach 2	T6: [X1] This means [Z1]. $[X1]$ This does not mean [Z1]	T5-small	Naive Bayes	0.493	0.450
			SVM	0.535	0.480
			Decision Tree	0.543	0.498
			T5-small	0.845	0.678
			BERT-base	0.856	0.671
	T6: [X1] This means [Z1]. $[X1]$ This does not mean [Z1]	BART-large	Naive Bayes	0.502	0.527
			SVM	0.674	0.513
			Decision Tree	0.629	0.512
			T5-small	0.830	0.677
			BERT-base	0.892	0.712
	T7: The correct keypoint for the argument: "[X1]" is [Z1] The wrong keypoint for the argument: "[X1]" is [Z1]	T5-small	Naive Bayes	0.485	0.451
			SVM	0.573	0.499
			Decision Tree	0.549	0.501
			T5-small	0.835	0.698
			BERT-base	0.900	0.679
	T7: The correct keypoint for the argument: "[X1]" is [Z1] The wrong keypoint for the argument: "[X1]" is [Z1]	BART-large	Naive Bayes	0.500	0.520
			SVM	0.667	0.529
			Decision Tree	0.614	0.477
			T5-small	0.810	0.678
			BERT-base	0.896	0.664

Table 4: Results of baselines and Approach 1 and 2 for cross/in-domain experiments

for the in-domain experiment is minimal, with Approach 1 getting 91.4% and Approach 2 90.0%. We obtain a higher F1-score using Approach 2 (90%) compared to the baselines (88%) on the in-domain dataset, but on the cross-domain dataset, the F1-score from Approach 2 (69.8%) is lower than the

baseline (72.0%).

5.2 Error Analysis of Approach 2

Table 4 shows that the overall performance of Approach 2 is poor in comparison with Approach 1 for cross/in-domain experiments. We dive into the reason hidden behind this result and make two assumptions. The first assumption is that the language models used for getting the intermediary texts suffer from negation issue. As explained in section 3.3.2, we use slightly different templates for matching/non-matching argument-keypoint pairs to generate their intermediary texts. The templates for non-matching pairs contain a negative connotation like `not` or `wrong`, but the problem is that the PLMs (T5-small and BART-large) cannot capture negation which is demonstrated by the generated intermediary texts being almost the same regardless of whether the template is positive or negative, and thus it results in lower F1-scores from Approach 2. To make it clear, the two following intermediary text examples are extracted from the train set. Given an argument and matching and non-matching key points corresponding to the argument, we can see that their intermediary texts are the same irrespective of using two versions of prompt templates (e.g. positive and negative).

- **Argument**: *by copying something you can not get a pure copy. each copy that is made is worse then the other meaning that no one knows what can happen with cloning.*
- **Keypoints**:
  matching - *Cloning is not understood enough yet*
  non-matching - *Cloning is unethical/anti-religious-*
- **Intermediary texts for both matching/non-matchin pairs**: *Cloning is unnatural*

The second assumption is a decision-making process during the evaluation time. Alluded to previously, the corresponding template is selected based on matching/non-matching labels for each pair in the train set to generate intermediary texts. However, the labels are hidden in the test set, and the specific template can not be chosen. On this account, we always use the non-negative templates (*This means* and *The correct key point*) to get the intermediary text during the inference time.

Even though the overall results of Approach 2 are not as good as we expect, there are still some promising aspects. If the negation issue is solved successfully, Approach 2 could alleviate the need for predefined key points since it can automatically generate texts/key points.

6 Conclusions and Future Work

In this work, we first build the baseline models for the argument to keypoint mapping task by finetuning PLMs without implementing prompt engineering. Then, we take advantage of prompt-based learning and utilize it while finetuning PLMs with two different approaches. From the comparison between the baselines and the two specific approaches, prompt engineering substantially improves the performance of the task. However, it still includes some challenges and limitations that need to be investigated more in the case of Approach 2. To be more specific, in Approach 1, we attempt five different prompt templates with T5-base, and all of the results are better than the baselines for both cross/in-domain experiments. While in Approach 2, the intermediary texts generated from T5-small and BART-large using two prompt templates reduce the overall performance compared to baselines.

The mapping of arguments to key points can be viewed as an intermediate step toward fully automatic argument summarization. Therefore, in future work, we plan to tackle the negation problem of PLMs in Approach 2, which would be promising for generating key points automatically. Furthermore, experimenting with other sequence-to-sequence models using prompt-based learning is another interesting future direction.

Acknowledgement

The authors would like to thank Dr. Khalid Al-Khatib of the University of Groningen, The Netherlands for his support and assistance.

References

Bar-Haim, R., L. Eden, R. Friedman, Y. Kantor, D. Lahav, and N. Slonim (2020). From arguments to key points: Towards automatic argument summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, pp. 4029–4039. Association for Computational Linguistics.

Bar-Haim, R., Y. Kantor, L. Eden, R. Friedman, D. Lahav, and N. Slonim (2020). Quantitative argument summarization and beyond: Cross-domain key point

analysis. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, pp. 39–49. Association for Computational Linguistics.

Bird, S., E. Klein, and E. Loper (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Brown, T., B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems 33*, 1877–1901.

Cortes, C. and V. Vapnik (1995). Support-vector networks. *Machine learning 20*(3), 273–297.

Devlin, J., M. Chang, K. Lee, and K. Toutanova (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR abs/1810.04805*.

Ding, N., S. Hu, W. Zhao, Y. Chen, Z. Liu, H.-T. Zheng, and M. Sun (2021). Openprompt: An open-source framework for prompt-learning. *arXiv preprint arXiv:2111.01998*.

Gretz, S., R. Friedman, E. Cohen-Karlik, A. Toledo, D. Lahav, R. Aharonov, and N. Slonim (2020). A large-scale dataset for argument quality ranking: Construction and analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Volume 34, pp. 7805–7813.

Huang, D., L. Cui, S. Yang, G. Bao, K. Wang, J. Xie, and Y. Zhang (2020, November). What have we achieved on text summarization? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, pp. 446–469. Association for Computational Linguistics.

Jiang, Z., A. Anastasopoulos, J. Araki, H. Ding, and G. Neubig (2020, November). X-FACTR: Multilingual factual knowledge retrieval from pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, pp. 5943–5959. Association for Computational Linguistics.

Kapadnis, M., S. Patnaik, S. Panigrahi, V. Madhavan, and A. Nandy (2021, November). Team enigma at ArgMining-EMNLP 2021: Leveraging pre-trained language models for key point matching. In *Proceedings of the 8th Workshop on Argument Mining*, Punta Cana, Dominican Republic, pp. 200–205. Association for Computational Linguistics.

Lan, Z., M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut (2020). Albert: A lite bert for self-supervised learning of language representations. In *ICLR*. OpenReview.net.

Lewis, M., Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer (2020, July). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, pp. 7871–7880. Association for Computational Linguistics.

Liu, P., W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig (2021). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ArXiv abs/2107.13586*.

Liu, Y., M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov (2019). Roberta: A robustly optimized bert pretraining approach. *ArXiv abs/1907.11692*.

McCallum, A., K. Nigam, et al. (1998). A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, Volume 752, pp. 41–48. Citeseer.

Quinlan, J. R. (1986, March). Induction of decision trees. *Mach. Learn. 1*(1), 81–106.

Radford, A., K. Narasimhan, T. Salimans, and I. Sutskever (2018). Improving language understanding by generative pre-training.

Raffel, C., N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research 21*(140), 1–67.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin (2017). Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, Volume 30. Curran Associates, Inc.

Yang, Z., Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, Volume 32. Curran Associates, Inc.

Yuan, W., G. Neubig, and P. Liu (2021). Bartscore: Evaluating generated text as text generation. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan (Eds.), *Advances in Neural Information Processing Systems*, Volume 34, pp. 27263–27277. Curran Associates, Inc.