Matching The Statements: A Simple and Accurate Model for Key Point Analysis

Viet Hoang Phan **Tien Long Nguyen** **Duc Long Nguyen** **Ngoc Khanh Doan**
School of Information and Communication Technology
Hanoi University of Science and Technology
{hoang.pv180086, long.nt180129, long.nd183583,
khanh.dn180110}@sis.hust.edu.vn

Abstract

Key Point Analysis (KPA) is one of the most essential tasks in building an Opinion Summarization system, which is capable of generating key points for a collection of arguments toward a particular topic. Furthermore, KPA allows quantifying the coverage of each summary by counting its matched arguments. With the aim of creating high-quality summaries, it is necessary to have an in-depth understanding of each individual argument as well as its universal semantic in a specified context. In this paper, we introduce a promising model, named Matching the Statements (MTS) that incorporates the discussed topic information into arguments/key points comprehension to fully understand their meanings, thus accurately performing ranking and retrieving best-match key points for an input argument. Our approach[1] has achieved the 4^{th} place in Track 1 of the Quantitative Summarization – Key Point Analysis Shared Task by IBM, yielding a competitive performance of 0.8956 (3^{rd}) and 0.9632 (7^{th}) strict and relaxed mean Average Precision, respectively.

1 Introduction

Prior work in Opinion Summarization often followed the extractive strategy, which identifies the most representative pieces of information from the source text and copies them verbatim to serve as summaries (Angelidis and Lapata, 2018; Brazinskas et al., 2019). Abstractive summarization is a less popular strategy compared to the previous one yet offers more coherent output texts. Approaches governed by this vein could generate new phrases, sentences or even paragraphs that may not appear in the input documents (Ganesan et al., 2010; Isonuma et al., 2021). Both extractive and abstractive methods are the straightforward applications of multi-document summarization (Liu et al., 2018; Fabbri et al., 2019), which has been an emerging research domain of natural language processing in recent years.

As is well known, in traditional multi-document summarization methods, the role of an individual or a subset of key points among the summaries is often neglected. To be more specific, Bar-Haim et al. (2020) posed a question regarding the summarized ability of a small group of key points, and to some extent answered that question on their own by developing baseline models that could produce a concise bullet-like summary for the crowd-contributed arguments. With a pre-defined list of summaries, this task is known as Key Point Matching (KPM). Figure 1 provides a simple illustration of the KPM problem, where the most relevant key points are retrieved for each given argument within a certain topic (i.e. context).

Inspired by the previous work that studied the problem of learning sentence representation (Cer et al., 2018; Reimers and Gurevych, 2019) and semantic similarity (Yan et al., 2021), we propose Matching The Statements (MTS), which further takes the topic information into account and effectively utilizes such proper features to learn a high performance unified model. Our approach has benefited from the recent developments of pre-trained language models such as BERT (Devlin et al., 2018), ALBERT (Lan et al., 2019) or RoBERTa (Liu et al., 2019).

Our contributions in this paper could be depicted as follows:

- Firstly, we design a simple yet efficient network architecture to fuse the context into sentence-level representations. Instead of letting the model infer the implicit reasoning structure, we provide our model with the information on whether an argument or key point (which are collectively referred to as statements in the remainder of this paper) supports its main topic or not.

[1]The code is available at: https://github.com/VietHoang1512/KPA

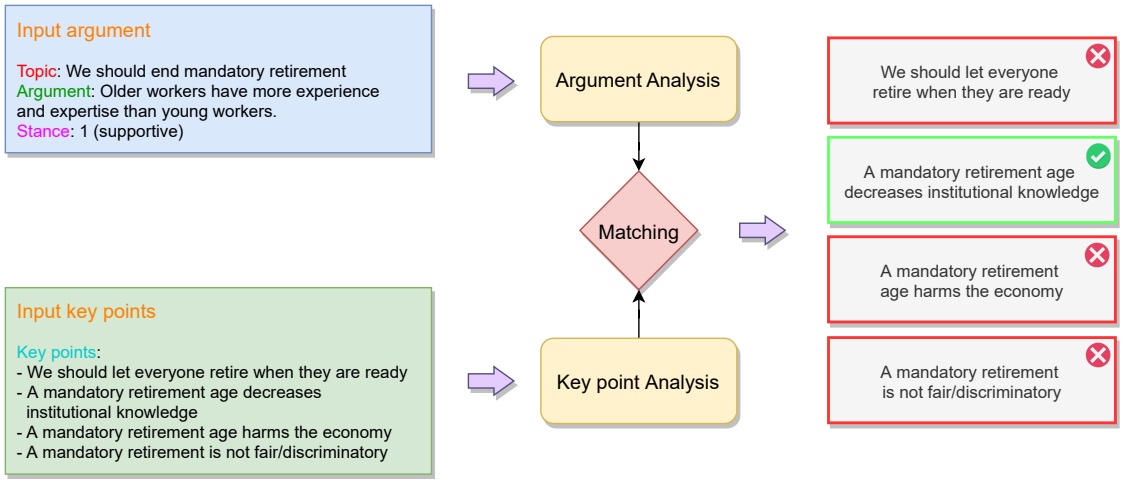

Figure 1: Overview of the Key Point Matching workflow in the Quantitative Summarization – Key Point Analysis Shared Task Track 1. From the information retrieval perspective, this task is to identify the most salient point that reinforces a given query.

- Secondly, our method adopts the pseudo-labels mechanism (Ge et al., 2020; Zhang et al., 2021), where we label arguments that belong to the same key point (and the key point itself) by the same index. The goal is to learn an embedding space in which the embedded vectors of mutual supportive statement pairs (i.e. having the same label) are pulled closer whereas unrelated ones are pushed apart.

- Finally, we validate the proposed MTS on the ArgKP-2021 (Bar-Haim et al., 2020) dataset in a variety of protocols. Extensive experiment results show that our proposed method strongly outperforms other baselines without using external data, thus becoming a potential method for the Key Point Matching problem.

The rest of this paper is organized in the following way: Section 2 briefly reviews the related work, while section 3 formulates the KPM problem. Next, we describe our methodology in section 4, followed by the experimental results in section 5. Finally, section 6 will conclude our work and discuss future directions for further improvements.

2 Related Work

A standard approach for key points and arguments analysis is properly extracting their meaningful semantics. Our model stems from recent literatures that are based on siamese neural networks (Reimers and Gurevych, 2019; Gao et al., 2021) to measure the semantic similarity between documents. Even though, MTS has its own unique characteristics to incorporate context information.

2.1 Sentence Embeddings

The representation of sentences in a fixed-dimensional vector space plays a crucial role in enhancing a model's performance on downstream tasks. Early methods relied on static word embeddings (Pennington et al., 2014; Bojanowski et al., 2017), which encoded a sentence by directly averaging its word vectors or employing recurrent neural network (RNN) encoders (Conneau et al., 2017) and taking the pooled output from the hidden units. Despite the fact that these methods can leverage both syntactic and semantic features, they often fail to retain the contextual information or suffer from slow training (due to the sequential nature of RNNs).

That is where BERT (Devlin et al., 2018) as well as its variants come in and dominate the modern NLP research. Training these architectures can exploit the parallel computational capacity of GPUs/TPUs hardware accelerators. In SBERT, Reimers and Gurevych (2019) proposed a sentence embedding method via fine-tuning BERT models on natural language inference (NLI) datasets. More recent studies in learning sentence representation followed the contrastive learning paradigm and achieved state-of-the-art performance on numerous of benchmark tasks (Liao, 2021; Yan et al., 2021).

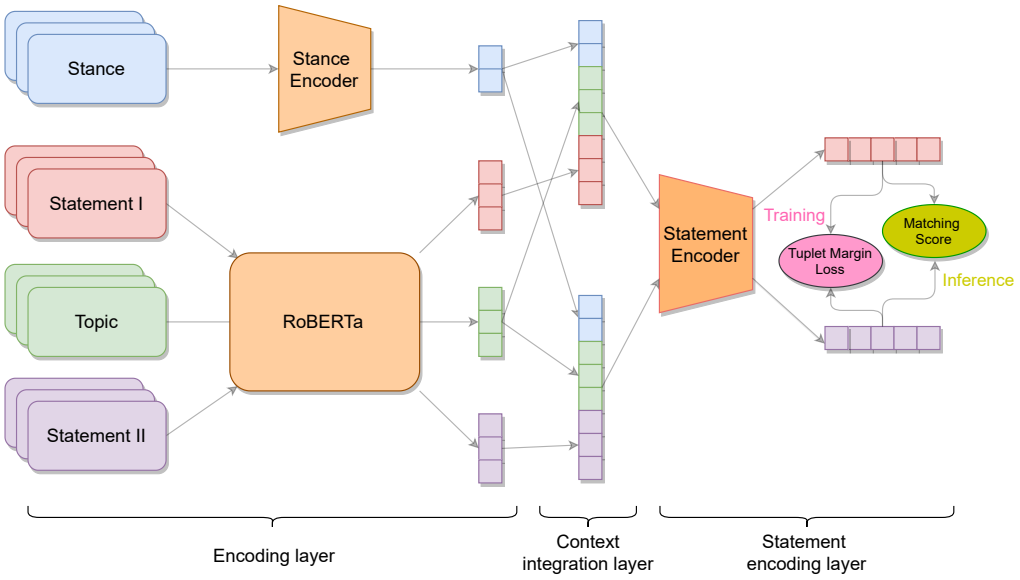

Figure 2: The overall design of our Matching The Statements architecture.

2.2 Semantic matching

Semantic matching is a long-standing problem and has a wide range of applications, such as: question-answering (Yang et al., 2015), text summarization (Zhong et al., 2020) and especially, information retrieval (Huang et al., 2013; Guo et al., 2016). Jiang et al. (2019) introduced a hierarchical recurrent neural network that could capture long-term dependency and synthesize information from different granularities (i.e. words, sentences or paragraphs). Similarly, Yang et al. (2020) replaced the RNN backbones with transformer-based models and modified self-attention architectures to adapt with long document inputs.

However, most of the existing work focuses only on assessing the similarity between pairs of sentences without paying attention to their context - which can help the reader to get an overview of the discussed topic. Recently, the ArgKP-2021 dataset has been published by Bar-Haim et al. (2020), which consists of annotations about whether two statements and their stances towards a specific topic match or not. The next sections will provide an overview of this dataset and how our model is applicable in the Quantitative Summarization – Key Point Analysis Shared Task [2].

3 Problem definition

In this shared task, we are provided with a dataset consisting of 28 different topics. Each topic contains a set of associated arguments and key points in the form of matching (with label 1) or non-matching (with label 0) pairs. The stances of these statements (whether a claim agrees or disagrees with its topic) are also exposed, we further evaluate the impact of this information in section 5.6.2.

In short, the Key Point Matching problem is formulated as follows: Given a controversial topic T with a list of m arguments and n key points $A_1, A_2, \ldots, A_m$; $K_1, K_2, \ldots, K_n$, along with their corresponding stances $S_1, S_2, \ldots, S_{m+n}$ $(S_i \in \{-1, 1\})$, which imply the attack or support relationships against the topic, our task is to rank key points that have the same stance with an input argument by the matching score. This priority is dependent on both the topic and the semantic of statements.

4 Methodology

The proposed MTS architecture is graphically shown in figure 2. It takes four separate inputs: (i) discussed topic, (ii) first statement, (iii) second statement, and (iv) their stance toward the topic. The final output is the similarity score of the fed in statements with respect to the main context. In the remainder of this section, we would like to describe three main components of MTS: encoding, context integration and statement encoding layers.

4.1 Data preparation

We observe that a small percentage of the arguments (4.71%) belong to two or more key points,

[2] https://2021.argmining.org/shared_task_ibm.html

while the rest are matched with at most one. For that reason, a straightforward idea is gathering arguments, which belong to the same key point, and label the clusters in order. In other words, each cluster is represented by a key point K_i, contains K_i and its matching arguments. Our clustering technique results in the fact that there are a small number of arguments that belong to multiple clusters. Arguments that do not match any of the key points are grouped into the NON-MATCH set.

Intuitively, if two different arguments support the same key point, they tend to convey similar meanings and should be considered as a matching pair of statements. Conversely, statements from different clusters are considered non-match in our approach. This pseudo-label method thus utilizes the similar semantic of within-cluster documents and enhances the model robustness. In the remainder of this paper, those arguments that come from the same cluster are referred to as positive pairs, otherwise, they are negative pairs.

During training, we use each key point and its matching/non-matching arguments (based on the annotation in the ArgKP-2021 dataset) in a mini-batch. Moreover, we also sample a small proportion of the NON-MATCH arguments and merge them into the mini-batch. Specifically, all the NON-MATCH arguments are considered to come from different and novel clusters. Because the definition of positive/negative statement pairs is well-defined, we can easily compute the loss in each mini-batch with a usual metric learning loss (Chopra et al., 2005; Yu and Tao, 2019).

4.2 Encoding layer

We first extract the contextualized representation for textual inputs using the RoBERTa (Liu et al., 2019) model. We adopt a canonical method (Sun et al., 2019) to achieve the final embedding of a given input, which is concatenating the last four hidden states of the [CLS] token. These embeddings are fed into the context integration layer as an aggregate representation for topics, arguments and key points. For example, a statement vector at this point is denoted as [3]:

$$\begin{aligned}\mathbf{h}^X &= [\, h_1^X, h_2^X, \ldots, h_{4\times 768}^X \,] \;\; (h_i^X \in \mathbb{R}) \\ &= [\, h_1^X, h_2^X, \ldots, h_{3072}^X \,]\end{aligned}$$

[3]For a consistent notation, statements and stances are denoted by uppercase letters: X and S

with 768 is the number of hidden layers produced by the RoBERTa-base model.

For the stance encoding, we employ a fully-connected network with no activation function to map the scalar input to a N-dimensional vector space. The representation of each topic, statement and stance are denoted as $\mathbf{h}^T$, $\mathbf{h}^X$ and $\mathbf{h}^S$, respectively.

4.3 Context integration layer

After using the RoBERTa backbone and a shallow neural network to extract the embeddings acquired from multiple inputs, we conduct a simple concatenation with the aim of incorporating the topic (i.e. context) and stance information into its argument/key point representations. After this step, the obtained vector for each statement is ([;] notation indicates the concatenation operator):

$$\mathbf{v}^X = [\mathbf{h}^S; \mathbf{h}^T; \mathbf{h}^X]$$

where $\mathbf{v}^X \in \mathbb{R}^{N+2\times 3072}$

4.4 Statement encoding layer

The statement encoding component has another fully-connected network on top of the context integration layer to get the final D-dimensional embeddings for key points or arguments:

$$\mathbf{e}^X = \mathbf{v}^X\,\mathbf{W} + \mathbf{b}$$

where $\mathbf{W} \in \mathbb{R}^{(N+6144)\times D}$ and $\mathbf{b} \in \mathbb{R}^D$ are the weight and bias parameters.

Concretely, training our model is equivalent to learning a function $f(S, T, X)$ that maps the similar statements onto close points and dissimilar ones onto distant points in $\mathbb{R}^{(N+6144)\times D}$.

4.5 Training

In each iteration, we consider each input statement from the incoming mini-batch as an anchor document and sample positive/negative documents from within/inter clusters. For calculating the matching score between two statements, we compute the cosine distance of their embeddings:

$$\begin{aligned}\mathcal{D}_{\text{cosine}}(\mathbf{e}^{X_1}, \mathbf{e}^{X_2}) &= 1 - \cos(\mathbf{e}^{X_1}, \mathbf{e}^{X_2}) \\ &= 1 - \frac{{\mathbf{e}^{X_1}}^T \mathbf{e}^{X_2}}{||\mathbf{e}^{X_1}||_2\, ||\mathbf{e}^{X_2}||_2}\end{aligned} \quad (1)$$

Empirical results show that cosine distance yields the best performance compared to Manhattan distance ($||\mathbf{e}^{X_1} - \mathbf{e}^{X_2}||_1$) and Euclidean distance ($||\mathbf{e}^{X_1} - \mathbf{e}^{X_2}||_2$). Hence, we use cosine as

the default distance metric throughout our experiments. We also revisit several loss functions, such as contrastive loss (Chopra et al., 2005), triplet loss (Dong and Shen, 2018) and tuplet margin loss (Yu and Tao, 2019). Unlike previous work, Yu and Tao (2019) use another distance metric, which will be described below.

Assume that a mini-batch consists of $k+1$ samples $\{X_a, X_p, X_{n_1}, X_{n_2}, \ldots, X_{n_{k-1}}\}$, which satisfies the tuplet constraint: X_p is a positive statement whereas X_{n_i} are X_a's negative statements w.r.t X_a. Mathematically, the tuplet margin loss function is defined as:

$$\mathcal{L}_{\text{tuplet}} = \log(1 + \sum_{i=1}^{k-1} \mathrm{e}^{s(\cos\theta_{an_i} - \cos(\theta_{ap} - \beta))})$$

where θ_{ap} is the angle between e^{X_a} and e^{X_p}; θ_{an_i} is the angle between e^{X_a} and $e^{X_{n_i}}$. β is the margin hyper-parameter, which imposes the distance between negative pair to be larger than β. Finally, s acts like a scaling factor.

Additionally, Yu and Tao (2019) also introduced the intra-pair variance loss, which was theoretically proven to mitigate the intra-pair variation and improve the generalizability. In MTS, we use a weighted combination of both tuplet margin and intra-pair variance as our loss function. The formulation of the latter one is:

$$\mathcal{L}_{pos} = \mathbb{E}[(1-\epsilon)\,\mathbb{E}[\cos\theta_{ap}] - \cos\theta_{ap}]_+^2$$
$$\mathcal{L}_{neg} = \mathbb{E}[\cos\theta_{an} - (1+\epsilon)\,\mathbb{E}[\cos\theta_{an}]_+^2$$
$$\mathcal{L}_{\text{intra-pair}} = \mathcal{L}_{pos} + \mathcal{L}_{neg}$$

where $[\cdot]_+ = \max(0, \cdot)$.

As pointed out by Hermans et al. (2017); Wu et al. (2017), training these siamese neural networks raises a couple of issues regarding easy/uninformative examples bias. In fact, if we keep feeding random pairs, more easy ones are included and prevent models from training. Hence, a hard mining strategy becomes crucial for avoiding learning from such redundant pairs. In MTS, we adapt the multi-similarity mining from Wang et al. (2019), which identifies a sample's hard pairs using its neighbors.

Given a pre-defined threshold ϵ, we select the negative pairs if they have the cosine similarity greater than the hardest positive pair, minus ϵ. For instance, let X_a be a statement, which has its positive and negative sets of statements denoted by $\mathcal{P}_{X_a}$ and $\mathcal{N}_{X_a}$, respectively. A negative pair of statements $\{X_a, X_n\}$ is chosen if:

$$\text{cosine}(\mathbf{e}^{X_a}, \mathbf{e}^{X_n}) \geq \min_{X_i \in \mathcal{P}_{X_a}} \text{cosine}(\mathbf{e}^{X_a}, \mathbf{e}^{X_i}) - \epsilon$$

Such pairs are referred to as hard negative pairs, we carry out a similar process to form hard positive pairs. If a positive pair $\{X_a, X_p\}$ is selected, then:

$$\text{cosine}(\mathbf{e}^{X_a}, \mathbf{e}^{X_p}) \leq \max_{X_i \in \mathcal{N}_{X_a}} \text{cosine}(\mathbf{e}^{X_a}, \mathbf{e}^{X_i}) + \epsilon$$

4.6 Inference

At inference time, we pair up the arguments and key points that debate on a topic under the same stance. Afterward, we compute the matching score based on the angle between their embeddings. For instance, an argument A and key point K will have a matching score of:

$$\begin{aligned}\text{score}(\mathbf{e}^A, \mathbf{e}^K) &= 1 - \mathcal{D}_{\text{cosine}}(\mathbf{e}^A, \mathbf{e}^K) \\ &= \cos(\mathbf{e}^A, \mathbf{e}^K)\end{aligned}$$

The right-hand side function squashes the score into the probability interval of $[0, 1)$ and compatible with the presented loss function in section 4.5.

5 Experiment

To verify the effectiveness of the Matching The Statements model, we conduct extensive experiments on the ArgKP-2021 (Bar-Haim et al., 2020) dataset and compare the performance of MTS against baselines.

5.1 ArgKP-2021 Dataset

ArgKP-2021 (Bar-Haim et al., 2020), the data set used in the Quantitative Summarization – Key Point Analysis Shared Task, is split into training and development sets with the ratio of $24:4$. The training set is composed of 5583 arguments and 207 key points while those figures in the development set are 932 and 36. Each argument could be matched to one or more key points, yet the latter ones account for a small proportion of the data, as stated in section 4.1. The texts presented in ArgKP-2021 are relatively short, with approximately 18.22 ± 7.76 by words or 108.20 ± 43.51 by characters.

5.2 Evaluation protocol

For evaluation, only the most likely key point is chosen for each argument based on the predicted

scores. These pairs are then sorted by their matching scores in descending order, and only the first half of them are included in the assessment. According to Friedman et al. (2021), there are two metrics used in Track 1, namely relaxed and strict mean Average Precision (mAP):

$$\textbf{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

Since there are some argument-key point pairs in the ArgKP-2021 dataset that have indecisive annotations (i.e. their label is neither matched nor non-matched): in the relaxed mAP evaluation, these pairs are considered as matched whereas strict mAP treats them as non-matched pairs.

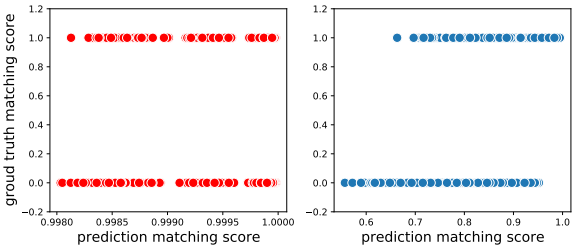

Figure 3: Statement representation before (left) and after (right) training.

5.3 Embeddings quality

Figure 3 depicts the qualitative representation learning result of MTS before and after training. In the beginning, the similarity scores between matched/non-match key point-argument pairs are extremely high (≈ 0.999). That means, almost all the statements are projected into a small region of the embedding space, and it is difficult to derive a cut-off threshold to get rid of the non-matching pairs.

Therefore, the mean Average Precision scores when we directly use the untrained model with RoBERTa backbone are relatively low. Though, our training procedure improves the model's distinguishability and reduces the collapsed representation phenomenon. Indeed, the similarity scores at this point are stretched out and the mAP scores significantly increase (strict mAP $0.45 \rightarrow 0.84$; relaxed mAP $0.62 \rightarrow 0.94$).

5.4 Baselines

For performance benchmarking, we implement two different baselines and their variations, namely Simple Argument-Key point matching (SimAKP)

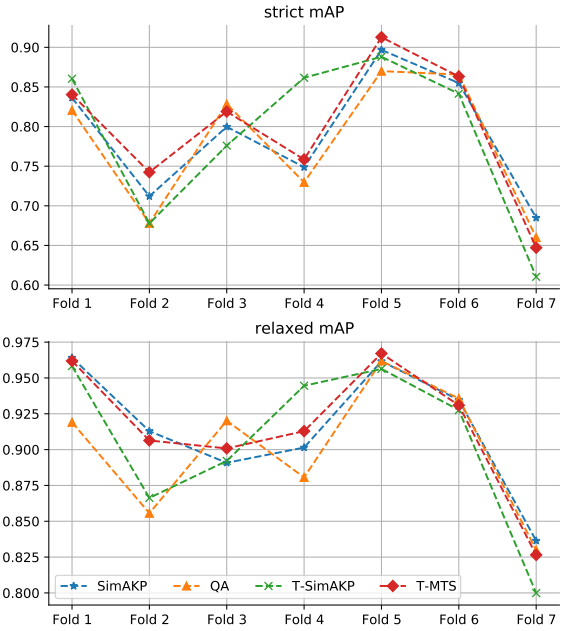

Figure 4: Mean Average Precision scores over 7 folds. The "T-" prefix denotes the models that use triplet loss (Dong and Shen, 2018) while the rest are trained with the contrastive loss (Chopra et al., 2005).

and Question Answering-liked Argument-Key point matching (QA) models. We construct a sampling strategy in an online manner: in each mini-batch, we select the hardest positive/negative pairs according to the method discussed in Section 4.5 to compute the loss.

Simple Argument-Key point matching: The architecture of SimAKP is the same as MTS with the main difference in the data preparation. Instead of clustering similar statements, SimAKP simply performs pair-wise classification on the ArgKP-2021 dataset. Equivalently, each input to the SimAKP model consists of an argument-key point pair. This approach will not make use of the analogous nature of these claims that matched with the same key point.

Question Answering-liked Argument-Key point matching: Inspired by the Question Answering, we format the arguments and key points fed to the RoBERTa model in order to incorporate the context into statements as below:

[CLS] Topic [SEP] [SEP] Statement [SEP]
[CLS] Topic [SEP] [SEP] Key point [SEP]

where [CLS] and [SEP] are two special tokens.

In particular, obtained outputs of RoBERTa model with the above inputs are then concatenated with the stance representations to produce a tensor with shape $(\text{batch size}, N + 3072)$, which is fed

to a fully connected layer to embed the semantic meaning of each individual statement.

5.5 Results

To facilitate evaluation, we set up a 7-fold cross-validation, each contains 24 topics for training and 4 topics for development. The train-dev split in Track 1 of Quantitative Summarization – Key Point Analysis Shared Task is replicated in fold 1.

As can be seen in Figure 4, we observe that our proposed MTS (we use triplet loss for a fair comparison) consistently outperforms other baselines in both mAP scores (higher is better). It achieved competitive scores on all splits, except fold 7. The reason is that the number of labeled argument-key point pairs of the development set in this part is the smallest among 7 folds, and there are substantial drops in terms of performance for all baselines.

Model	strict mAP	relaxed mAP
SimAKP	0.790 ± 0.072	0.914 ± 0.041
SimAKP w.o mining	0.783 ± 0.074	0.917 ± 0.037
T-SimAKP	0.788 ± 0.098	0.906 ± 0.054
T-SimAKP w.o mining	0.782 ± 0.101	0.901 ± 0.076

Table 1: The effect of hard sample mining in baselines.

We also examine the impact of hard negative mining in Table 1, the baselines are compared against themselves when using the hard mining strategy (i.e. avoid learning the embeddings of trivial samples). With the employment of hard mining, there is an improvement in performance for most baselines. Except for a small decrease in terms of relaxed mAP in SimAKP, both contrastive and triplet loss Simple Argument-Key point matching models have an average increase of 0.005% in mAP scores.

5.6 Differential Analysis

To provide insight analysis in the setting of Matching The Statements, we therefore create four different setups: original MTS, MTS with batch normalization (Ioffe and Szegedy, 2015) immediately after context integration layer, MTS without mining strategy, and triplet-loss MTS. Although tuplet margin loss has an up/down weighting hard/easy samples mechanism, we find that MTS with multi-similarity mining (Wang et al., 2019) performed best during the exploratory phase.

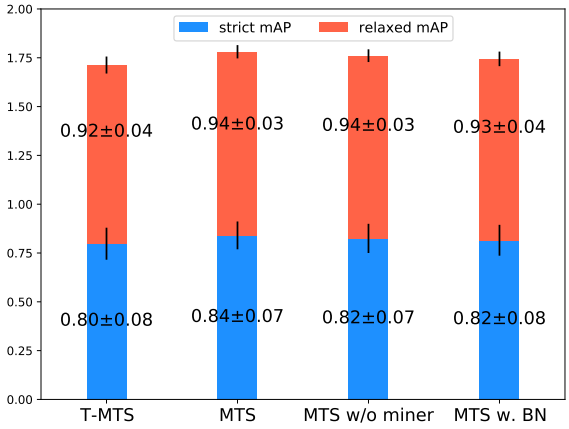

Figure 5: Switching off different setups shows that each component of the original MTS's setting contributes to its performance.

Figure 5 summarizes the average score for all setups. Overall, MTS performs similarly or better than its variants (without multi-similarity mining or adding a batch normalization layer). Replacing triplet loss with tuplet margin loss helps to boost both strict mAP and relaxed mAP by 0.2. Eventually, in an attempt to produce a consistent and accurate prediction on the test dataset, an ensemble of 4/7 best models from splits was used for final submission. As shown in Table 2, among the performances of the top-10 team, our proposed model achieved the third position in terms of strict mAP, 7^{th} position in relaxed mAP and 4^{th} overall.

#	Team	strict mAP	relaxed mAP
1	mspl	0.908 (2)	0.972 (3)
2	heinrichreimer	0.912 (1)	0.967 (5)
3	vund	0.878 (4)	0.968 (4)
4	**HKL (ours)**	**0.896 (3)**	**0.963 (7)**
5	sohanpatnaik	0.872 (5)	0.966 (6)
6	fengdoudou	0.853 (10)	0.98 (2)
7	mozhiwen	0.833 (12)	0.985 (1)
8	Fibelkorn	0.869 (6	0.952 (10)
8	emanuele.c	0.868 (7)	0.956 (9)
10	niksss	0.858 (8)	0.95 (11)

Table 2: Leaderboard of the Track 1 Quantitative Summarization – Key Point Analysis.

5.6.1 BERT embeddings

Here, we showcase the benefit of taking the concatenation of the last four hidden state layers of the [CLS] token as the aggregate representation for the whole document. The first part of Table 3 is a clear

Embedding	strict mAP	relaxed mAP	#Param
Sum all tokens	0.834 ± 0.065	0.938 ± 0.037	
Mean all tokens	0.796 ± 0.068	0.916 ± 0.034	125M
[CLS] last hidden layer	0.823 ± 0.072	0.937 ± 0.038	
[CLS] 4 hidden layers	$\mathbf{0.840 \pm 0.071}$	$\mathbf{0.941 \pm 0.034}$	126M
LUKE	0.808 ± 0.096	0.926 ± 0.056	276M
ALBERT	0.748 ± 0.071	0.879 ± 0.044	13M
MPNet	0.839 ± 0.059	0.940 ± 0.029	111M
DistilBERT	0.724 ± 0.065	0.864 ± 0.058	68M
BERT (uncased)	0.746 ± 0.062	0.888 ± 0.035	110M
BERT (cased)	0.752 ± 0.073	0.883 ± 0.057	

Table 3: Comparison between different embedding strategies and pre-trained language models. In this experiment, we report the result of the base version.

proof for this advantage, using only the last hidden layer of [CLS] can hurt the overall performance. Likewise, the mean-pooling or summing up the token embeddings has worse results, compared to our method.

To show the generality and applicability of our proposed model, we retain the MTS configuration when experimenting with other transformers-based backbones, such as: BERT (Devlin et al., 2018), ALBERT (Lan et al., 2019), DistilBERT (Sanh et al., 2019), LUKE (Yamada et al., 2020) or MPNet (Song et al., 2020). According to the second part of Table 3, among six pre-trained language models, MPNet yields a comparable result with RoBERTa (≈ 0.84 & 0.94) while requiring 10% less number of parameters. We also note that, the increase in model size of Language Understanding with Knowledge-based Embeddings (LUKE) compared with RoBERTa results in unexpected performance reduction.

5.6.2 Stance effect

Up till now, we have almost finished the needed experiments to examine the effectiveness of our methodology. In this subsection, we further investigate the importance of the stance factor in building the MTS model by posing a question: *"How good is MTS when it has to predict the implicit relation between claims and topic"*. Since the topic information is incorporated in encoding the statements, so perhaps it is sufficient to learn meaningful representations, without explicitly providing the stance information.

By discarding the stance-involved components in MTS 2, the averaged result in 7 folds conceivably degrades to 0.741 ± 0.094 in strict mAP but rises up to 0.952 ± 0.019 in relaxed mAP. This is because each argument now can be matched with key points that have different stances. According to this exploration, an open challenge for future research is finding a better way to comprehend statements within a topic (i.e. let the model infer the stance itself). For instance, one could consider employing the attention mechanism between a topic and its arguments and key points to characterize the relationship between them.

6 Conclusion

In this paper, we present an efficient key point matching method based on supervised contrastive learning. We suppose that clustering the statements will be beneficial for the model training, and empirically verify this conclusion in the experiments. In addition, we found a simple and effective technique to encode these statements, and thus yields superior performance. In terms of model architecture, the components are carefully designed to ensure productivity. Results on Track 1 of Quantitative Summarization – Key Point Analysis show our method is a conceptually simple approach yet achieves promising performance.

References

Stefanos Angelidis and Mirella Lapata. 2018. Summarizing opinions: Aspect extraction meets sentiment prediction and they are both weakly supervised. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3675–3686, Brussels, Belgium. Association for Computational Linguistics.

Roy Bar-Haim, Lilach Eden, Roni Friedman, Yoav Kantor, Dan Lahav, and Noam Slonim. 2020. From

arguments to key points: Towards automatic argument summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4029–4039, Online. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Arthur Brazinskas, Mirella Lapata, and Ivan Titov. 2019. Unsupervised multi-document opinion summarization as copycat-review generation. *arXiv preprint arXiv:1911.02247*.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Xingping Dong and Jianbing Shen. 2018. Triplet loss in siamese network for object tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 459–474.

Alexander Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084, Florence, Italy. Association for Computational Linguistics.

Roni Friedman, Lena Dankin, Yoav Katz, Yufang Hou, and Noam Slonim. 2021. Overview of kpa-2021 shared task: Key point based quantitative summarization. In *Proceedings of the 8th Workshop on Argumentation Mining*. Association for Computational Linguistics.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: A graph based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 340–348, Beijing, China. Coling 2010 Organizing Committee.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Yixiao Ge, Dapeng Chen, and Hongsheng Li. 2020. Mutual mean-teaching: Pseudo label refinery for unsupervised domain adaptation on person re-identification. *arXiv preprint arXiv:2001.01526*.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM international on conference on information and knowledge management*, pages 55–64.

Alexander Hermans, Lucas Beyer, and Bastian Leibe. 2017. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR.

Masaru Isonuma, Junichiro Mori, Danushka Bollegala, and Ichiro Sakata. 2021. Unsupervised abstractive opinion summarization by generating sentences with tree-structured topic guidance. *arXiv preprint arXiv:2106.08007*.

Jyun-Yu Jiang, Mingyang Zhang, Cheng Li, Michael Bendersky, Nadav Golbandi, and Marc Najork. 2019. Semantic text matching for long-form documents. In *The World Wide Web Conference*, pages 795–806.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Danqi Liao. 2021. Sentence embeddings using supervised contrastive learning. *arXiv preprint arXiv:2106.04791*.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. *arXiv preprint arXiv:2004.09297*.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.

Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. 2019. Multi-similarity loss with general pair weighting for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5022–5030.

Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. 2017. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2840–2848.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. Luke: deep contextualized entity representations with entity-aware self-attention. *arXiv preprint arXiv:2010.01057*.

Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. Consert: A contrastive framework for self-supervised sentence representation transfer. *arXiv preprint arXiv:2105.11741*.

Liu Yang, Mingyang Zhang, Cheng Li, Michael Bendersky, and Marc Najork. 2020. Beyond 512 tokens: Siamese multi-depth transformer-based hierarchical encoder for long-form document matching. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1725–1734.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal. Association for Computational Linguistics.

Baosheng Yu and Dacheng Tao. 2019. Deep metric learning with tuplet margin loss. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6490–6499.

Xiao Zhang, Yixiao Ge, Yu Qiao, and Hongsheng Li. 2021. Refining pseudo labels with clustering consensus over generations for unsupervised object re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3436–3445.

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. Extractive summarization as text matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6197–6208, Online. Association for Computational Linguistics.