Argument Extraction for Key Point Generation Using MMR-Based Methods

DAIKI SHIRAFUJI¹, RAFAL RZEPKA², AND KENJI ARAKI², (Member, IEEE)
¹Graduate School of Information Science and Technology, Hokkaido University, Hokkaido 060-0814, Japan
²Faculty of Information Science and Technology, Hokkaido University, Hokkaido 060-0814, Japan
Corresponding author: Rafal Rzepka (rzepka@ist.hokudai.ac.jp)

ABSTRACT When people debate, they want to familiarize themselves with a whole range of arguments about a given topic in order to deepen their knowledge and inspire new claims. However, the amount of differently phrased arguments is humongous, making the process of processing them time-consuming. In spite of many works on using arguments (e.g. counter-argument generation), there is only a few studies on argument aggregation. To address this problem, we propose a new task in argument mining – Argument Extraction, which gathers similar arguments into key points, usually single sentences describing a set of arguments for a given debate topic. Such a short summary of related arguments has been manually created in previous research, while in our research key point generation becomes fully automatic, saving time and cost. As the first step of key point generation we explore existing similarity calculation methods, i.e. Sentence-BERT and MoverScore to investigate their performance. Next, we propose a combination of argument similarity and Maximal Marginal Relevance (MMR) for extracting key phrases to be utilized in our novel task of Argument Extraction. Experimental results show that MoverScore-based MMR outperforms strong baselines covering 72.5% of arguments when eleven or more arguments are extracted. This percentage is almost identical with the cover rate of human-made key points.

INDEX TERMS Argument aggregation, argument mining, key points, machine learning, natural language processing, text summarization.

I. INTRODUCTION
Arguments in debate or discussion sites are broadly studied in NLP area [1], [2]. Research field related to arguments is called argument mining, and this field has gained popularity during last ten years. In argument mining, the argument components (e.g. claims, premises) are basically extracted from unstructured texts or utterances [3]. These attempts for detecting argument components have been clarifying where is the main claim of an argument stated by an author (or a speaker), where is an evidence for the claim, and where an example for the argument is expressed. Currently, accuracy of distinguishing above-mentioned components has significantly improved, therefore targets of the area of argument mining have gradually shifted to different topics. Argument mining researchers have focused on predicting persuasiveness of an argument [4], [5], filling gaps between two claims [6], [7], and predicting argument stance (pro or con) [8], [9]. These works have benefited from a very large amount of argument data from debates, discussions, speeches, or essays. However, studies on aggregating such numerous arguments seem to be neglected. Big number of arguments makes it difficult to decide which one people commonly discuss. Although there are various and numerous studies in the field of argument mining, no method will be practical if users cannot find accurate opinions for their debating topic. There is a need to narrow down the well-discussed arguments, i.e. summarize a lot of arguments into some core arguments. To address this problem, plentiful arguments are needed, and a new question arises; how to gather them? There are typically two approaches to aggregating arguments; one way is to retrieve them from online debate resources (like iDebate¹ or CreateDebate²), another is to gather arguments from debate spectators who express their opinions [2]. Using both methods, the number of collected arguments in a

¹https://idebate.org
²https://www.createdebate.com
In order to address this problem, Reimers et al. [10] use capabilities of contextualized word embeddings of ELMo [11] and BERT [12] to classify and cluster topic-dependent arguments from Argument Facet Similarity Corpus [13]. They cluster related arguments, however they do not generate summaries of arguments. This problem is mentioned in Bar-Haim et al. [14], who defined the key point: usually a single sentence describing a set of similar arguments, i.e. a summary of related arguments (see Table 1). Bar-Haim et al. found that 14 key points can cover approximately 72.5% arguments in a debate topic. Two examples are shown in Table 2 with a debate topic, its stance, and the number of arguments included in the topic, its key point, and the ratio of how many arguments the key point covers. It can be said that key points are useful to represent arguments from these two examples. Therefore, Bar-Haim et al. created a large dataset consisting of [argument, key point] pairs. Next, two experimental steps are conducted: Match Scoring and Match Classification. In Match Scoring, a match score for a given [argument, key point] pair is computed, (see Table 1, which shows examples of the dataset for Match Scoring, named ArgKP). In Match Classification, the methods of Bar-Haim and colleagues discover matching key points for each argument. Authors report that in these tasks BERT-large [12] fine-tuned model achieves the best results in supervised learning, and a BERT-large embedding method performs best among unsupervised methods.

Although Bar-Haim et al. claim that their research utilizes argument summarization, only the similarity between key points and arguments is calculated, which means their both experiments are almost identical to the above-mentioned research of Reimers et al. [10]. Furthermore, for Match Scoring and Match Classification, manual creation of key points is necessary beforehand. This indicates that their methods cannot be used with new debate topics for which key points do not exist. Therefore, there is a need to automatically generate key points from a set of arguments in order to become applicable to novel debate topics.

In our paper, we focus on two tasks, an existing task Match Scoring and our proposed task Argument Extraction for the first step toward generating key points. To address these tasks, we define an argument as a main claim in utterances or texts like these represented in Table 1.

For achieving high performance in the Match Scoring task, we explore methods to calculate similarity between arguments and key points with the ArgKP dataset [14], which is explained in Section III-A. It is assumed that one of the important factors in summarizing large number of arguments into some key points is to cluster arguments into key points, not to select key points for one argument. Therefore, the Match Scoring, not the Match Classification, is adopted for our target task in this paper. The previous research [14] experimented with both supervised and unsupervised methods, however the task is tackled with an unsupervised approach because the size of the data is too small when the ArgKP dataset is utilized for Argument Extraction (the dataset size is only 56 sets (28 debate topics for each stance) as described in Section IV-A2). The previous unsupervised approaches only averaged word vectors for computing a sentence vector, so the weight of important word information is identical with other words during sentence vector calculation. To address this problem, in our paper we propose to solve it by applying two methods: SentenceBERT [15] and MoverScore [16]. It is hypothesized that sentence embedding models could be a viable solution to represent a whole sentence in a vector, and that output vector will be able to represent significant words in an argument. Therefore, variations of Sentence-BERT model, which is fine-tuned with NLI dataset, and is able to represent sentence embeddings, is utilized. Moreover, MoverScore is used in
addition to Sentence-BERT model. This metric measures text similarity using Word Mover’s Distance [17] in order to calculate similarity between an argument and a key point on the word level. To the best of authors’ knowledge, this is the first attempt to tackle weighting the important words with sentence embedding models and MoverScore metric for the Match Scoring task.

Finally, we propose one new task, Argument Extraction, in which key points from a set of arguments in one debate topic and in each stance (pro or con) are generated. The purpose of this task is to extract various arguments as key points for each stance in a debate topic. The dataset structure for Argument Extraction is shown in Figure 1.3 Red, blue, and black-colored arguments belong to key point 1, 2, and both (1 and 2), respectively. Arguments in purple belong to no key point. In Argument Extraction, we try to extract one argument in red and one in blue from a set of arguments in order to represent key points 1 and 2.

In order to extract arguments, we adopt an unsupervised model, i.e. TextRank, as a baseline. We utilize an argument similarity for our proposed method, therefore, the best performed unsupervised approach in Match Scoring, i.e. MoverScore-based one is adopted for our proposed method. The whole novel pipeline for this new task and managing to avoid supervision determine the core of the originality of our contribution. To the best of authors’ knowledge, although there is a study on extracting various arguments [18] there is no research on extracting various arguments.

The main contributions of this article are:
1) Pointing out that existing methods which average word embeddings to compute sentence embeddings, cannot weight important words in an argument; and experimentally proving that both Sentence-BERT and MoverScore methods outperform the existing unsupervised methods, and the MoverScore approach yields the best results in the Match Scoring task;
2) Proposing a new task, Argument Extraction, toward generating key points for arguments;
3) Providing baselines for Argument Extraction using a classical unsupervised summarization methods;
4) Proposing a argument similarity-based method for Argument Extraction and showing that a MoverScore-based algorithm for extracting arguments outperforms other methods.

II. RELATED WORK
In this section, we describe research related to this study. First subsection explains the history of the field and similar topics in the area of argument mining. The second subsection presents existing approaches to measure text similarity. The final subsection describes document summarization algorithms and summarization tasks in argument mining area.

A. ARGUMENT MINING
Argument mining has been started by researchers working on the area of argumentative zoning, which is clarifying argument components [3] for debates, discussions or student essays. Aharoni et al. [3] create a benchmark for argumentative zoning, including 2,683 arguments in 33 debate topics. Lippi and Torroni [19] propose SVM-based approach for detecting claims in controversial texts. Stab and Grevych [20] perform 76% in F1 value using Integer Linear Programming. Their model outperforms others in the different two types of discourse structures.

After argument structure had been gradually solved, several researchers begin to work on argument persuasiveness [4] or persuasiveness of the whole debate [21] in order to generate more convincing arguments. Stance prediction is also an active research topic in argument mining. Several works estimate whether a given argument is supporting a particular debate topic or not (pro and con) [8], [9]. In order to conduct research on arguments from several perspectives, various large argument datasets have been recently created [22], [23]. Stab et al. [22] built one of argument retrieval systems which is able to search arguments for any debate topic. Their system covers about 89% of arguments found in expert-curated lists of arguments from an online debate portal (ProCon4). Similarly, ArgKP dataset created by Bar-Haim et al. [14] indicates high agreement between expert

3Data samples in this paper are quoted with the original capitalization and spelling (sentences were not corrected).

4https://www.procon.org
dataset (key point list) and arguments, achieving Cohen’s kappa = 0.82.

More recently, Misra et al. [13] attempt to measure a similarity of arguments in debate portals (iDebate and ProCon). Therefore, they provide Argument Facet Similarity corpus using these two debate portals. The purpose of their work is to identify whether an argument has the same facets as other arguments in across multiple debates. Several studies deal with facets using logical rules or contextual embedding models [10], [24]. These works are similar to mapping arguments to key points, but they do not calculate similarity between a facet and an argument. Moreover, they do not clarify key points, and in this aspect, their work clearly differs from our paper.

As for mapping arguments to key points, i.e. Match Scoring and Match Classification tasks, existing works [14], [25] can be given as most known examples. Boltužić and Šnajder [25] implement argument clustering. They map arguments derived from one online debate portal (iDebate) to arguments of another portal (iDebate). However, they work only on two debate topics with less than 400 arguments. Bar-Haim et al. [14] make a massive dataset including pairs of an argument and a key point labelled with relevancy. Moreover, they propose Match Scoring and Match Classification as the first steps to summarize arguments.

B. ARGUMENT MINING AND TEXT SIMILARITY

NLP researchers have been recently utilizing vectors of words / documents for measuring similarity between texts. Today, word vectors are usually computed with word embedding, sentence embedding, or language models. One of the most well-known algorithms, word2vec [26] is a pre-trained skip-gram or Continuous Bag-of-Words (CBOW) model to represent words as vectors. Following the pre-trained word embedding models, sentence and document embedding models have been also developed. For example, Skip-Thought Vectors [27] are based on a sentence encoder that predicts the surrounding sentences of a given sentence. Skip-Thought is based on an encoder-decoder model, its encoder maps words to a sentence vector, and its decoder generates the surrounding sentences. Sentence Transformers [15] are proposed by Reimers, and Gurevych, and they are sentence embedding models for English texts based on Siamese or triplet networks [28], [29]. Reimers and Gurevych insert language models (like BERT [12]) into the network of Siamese or triplet networks. They call such a BERT-based Siamese or triplet networks “Sentence-BERT.”

There are also some other methods to compute text similarity. Word Mover’s Distance algorithm [17] calculates the distance between texts with word2vec, and introduces Earth Mover’s Distance [30] into NLP field. Inspired by the Word Mover’s Distance, Zhao et al. [16] investigate encoding systems to devise a metric that shows a high correlation with human judgment of text quality in summarization or machine translation tasks. They name their metric MoverScore, whose details are described in Section III-C3.

As for the argument mining research in NLP field, Misra et al. propose Argument Facet Similarity task to measure similarity of arguments which are constructed with single sentences [13]. Their final goal is similar to our research, i.e. generating summaries for multiple arguments. Arguments in debate topics are usually paraphrased, therefore debaters often make argument summaries which represent similar arguments. By detecting facets, arguments can be mapped into its corresponding facet, therefore the Argument Facet Similarity task would be useful for summarization. Thus, for the tasks we describe in this paper we utilize the above-mentioned Sentence-BERT which is pre-trained on the Argument Facet Similarity dataset.

To measure argument similarities, Misra et al. introduce the Linguistics Inquiry Word Count (LIWC) tool / word2vec into Ridge Regression (RR) and Support Vector Regression (SVR). The SVR method with LIWC tools achieves 0.59, 0.26 and 0.40, respectively at Pearson correlation in three debate topics: Gun Control, Gay Marriage and Death Penalty. The RR method with LIWC tools yields 0.50, 0.27 and 0.34, respectively in those debate topics. The average results are 0.42 and 0.37 for SVR and RR, respectively. Reimers and Gurevych propose Sentence-BERT [15] to measure argument facet similarities as mentioned above. As explained before, Sentence-BERT is a BERT-based Siamese or triplet network to embed sentences into vectors. Its creators compare the fine-tuned BERT models and the fine-tuned Sentence-BERT models. BERT-base, which is fine-tuned on Argument Facet Similarity dataset, yields 0.77, and BERT-large fine-tuned on the dataset obtains 0.79 at Pearson correlation. Sentence-BERT-base and Sentence-BERT-large achieve 0.77 and 0.78 at Pearson correlation, respectively.

However, the way to construct Sentence-BERT is to train it on high quality dataset of labeled sentence pairs. This leads to limiting its application to tasks where the number of labeled data is extremely small. Zhang et al. [31] problematize this difficult situation, and introduce an unsupervised method to produce sentence embeddings. They propose Info-Sentence BERT (IS-BERT), which adds a novel self-supervised learning objective based on mutual information maximization strategies in order to obtain meaningful sentence embeddings with an unsupervised approach. Their algorithm embeds sentences with BERT model and CNNs with different window sizes to get concatenated local n-gram token embeddings. A discriminator takes all pairs of sentence and token representations as inputs and identifies whether these two sentences in a given pair are captured from the same sentences or not.

IS-BERT model achieves 0.49 at Pearson correlation on the Argument Facet Similarity dataset. Although the supervised methods outperform IS-BERT results, in domains without sufficient labelled datasets, IS-BERT obtain competitive results. However, in debate topics, the Sentence-BERT models perform equally with IS-BERT, so in our paper we utilize Sentence-BERT models instead of IS-BERT.
For the reasons described above, our proposed methods for Match Scoring are based on MoverScore and Sentence-BERT.

C. ARGUMENT MINING AND SUMMARIZATION

Non-neural approaches to summarization are traditionally based on extractive methods [32]–[35]. Even if there are now many neural methods to summarize texts, those non-neural ones are still used [36]. In particular, Maximal Marginal Relevance (MMR) [32], LexRank [33] and TextRank [35] are recently the most frequently used methods for comparison. MMR is also sometimes adopted in neural network-based methods [37]–[39]. One of these algorithms, EmbedRank [37], is an unsupervised approach with sentence embeddings capable of extracting a key phrase from a single document. In addition, these methods are easy to adapt not only to single document summarization (SDS), but also to multi-document summarization (MDS).

There are also abstractive summarization techniques for multi-document summarization with small datasets [38], [40]. The abstractive methods are recently based on deep neural networks [39], [41]. More recently, language models dedicated to summarization have also been developed, for example PEGASUS and MARGE [42], [43].

Also in argument mining area, summarization has recently become a subject of interest [44]–[46].

Egan et al. [44] summarize arguments in political-domain debate corpus. They address the summarization with creating structured argument summaries, therefore it becomes visible what arguments are associated with each other. However, their summaries do not focus on which argument should be shown in the tree-structured summaries.

Wang and Ling [45] try to summarize several arguments into one summary by integrating information from an important subset of inputted arguments using an encoder-based method. However, their summaries seem to only show the stance of arguments. For example, one of their system output summary is “The death penalty deters crime.” in a debate topic “This House supports the death penalty.” Such summaries could be misleading.

Alshomary et al. [46] summarize arguments which are constructed with several sentences. For each argument, they create the main claim automatically with their proposed PageRank-based abstractive summarization approach.

Recently, there is one work which focuses on generating key points for arguments by Bar-Haim et al. [18]. They extract high quality arguments as key point candidates with language models. Their study seems similar to our research, but their goal is to select high-quality arguments. We aim to obtain a variety of different arguments and avoid extracting similar ones. Moreover, they extract arguments with supervised approaches, while all of our methods are unsupervised.

In our research, as described above, we decided to use an unsupervised approach to avoid the dataset size problem. We choose MMR for our proposed method in order to extract various arguments. The reason we do not use transfer learning methods for language models like Bar-Haim et al. [18] did, is to avoid fine-tuning, which is one of the supervised approaches.

III. PRELIMINARY EXPERIMENT: MATCH SCORING

In this section, we introduce our preliminary experiments for finding adequate unsupervised methods to compute argument similarity for the proposed method in the Argument Extraction task.

In Match Scoring, we compute similarity in [argument, key point] pairs. Examples of matching and not matching pairs are shown in Table 1, which shows their contents, their stance to a given topic, and the manually annotated matching label. We try to identify whether such arguments and key points are similar or not. Match scores are evaluated with accuracy, precision, recall, and F1.

A. EXPERIMENTAL SETUP FOR MATCH SCORING

In this subsection, we describe the preparation for Match Scoring task: the task setting, and the data for our experiments.

1) TASK EXPLANATION OF MATCH SCORING

In Match Scoring, we compute similarity between an argument and a key point. Examples of matching and not matching pairs are given in Table 1, showing pairs, their stance to a given topic, and the manually annotated matching label. We try to identify exactly whether such arguments and key points are similar or not. We evaluate match scores with accuracy, precision, recall, and F1.

2) DATA FOR MATCH SCORING

We utilize ArgKP dataset [14] for Match Scoring. This dataset consists of 24,093 [argument, key point] pairs which are labeled as follows:

1) whether an argument and a key point describe the same content or not;
2) whether an argument represents pro side or con side.

To the best of our knowledge, ArgKP is the largest dataset usable for the Match Scoring. It is based on IBM-Rank-30k dataset [47] which contains 30,497 arguments annotated for their quality (from 0 to 1). Using the quality-indicating labels, Bar-Haim et al. [14] filter out unclear arguments whose quality score is lower than 0.5. Moreover, if argument polarity score is lower than 0.6, these arguments are also deleted in order to clarify the stances of arguments. After the arguments are filtered, debate experts, who are specialists in each debate topic domain, created key points for each debate topic (totally 28 topics) relevant to stances. These experts annotated the dataset according to the following instructions:

- Given a debate topic, generate a list of possible key points in a constrained time frame of 10 minutes [sic] per side.
- Experts should unify related key points that can be expressed as a single key point.
Out of the created key points, select a maximum of 7 per side that are estimated to be the most immediate ones, hence the most likely to be chosen by crowd workers.

Note that these three instructions are quoted (with original spelling) from the article which describes ArgKP dataset [14]. After this creation step, 378 key points were generated (6.75 per side per topic on average).

For the next step, eight annotators, excluding the debate experts who made key points, pair an argument with a key point. To ensure the quality of collected data, the following procedures were undertaken:

- Annotators answered whether each argument is pro or con toward each debate topic. If they fail to answer correctly in 10% or more arguments, all of their annotations are deleted.
- If the Cohen's Kappa between annotators is lower than 0.3, all annotator's results are discarded.
- All annotators are selected from workers who had performed well in the Bar-Haim et al.'s previous research.

In order to calculate inter-annotator agreements, [argument, key point] pairs with a binary label denoting whether the argument was matched to the key point are utilized. As a result, this dataset is regarded as having a "moderate agreement" because Fleiss' Kappa for this task was 0.44, and Cohen's Kappa was 0.5. Annotators marked all key points with associated arguments, and if no key point was relevant, they selected the "None" option.

In the end, 24,093 [argument, key point] pairs were generated, and Bar-Haim et al. named this dataset ArgKP.

Usually, arguments belong to one key point, but some of them correspond to two or three key points. There are only two arguments in the ArgKP dataset, which belong to three key points. One of them is "people whose quality of life is so poor that death would be preferable should be allowed to choose assisted suicide to end their suffering and pass with dignity", and their key points are "Assisted suicide gives dignity to the person that wants to commit it", "Assisted suicide reduces suffering", and "People should have the freedom to choose to end their life". Another argument relating to multiple key point is "Space exploration should be subsidized because the information learned from it could provide useful to our planet.", and its key points are "Space exploration improves science/technology", "Space exploration is necessary for the future survival of humanity", and "Space exploration unravels information about the universe".

In our research, we utilize all 24,093 [argument, key point] pairs from ArgKP dataset. We divide the dataset into 7 test topics and 21 training topics, following experimental setup of the previous research [14].

B. BASELINE METHODS FOR MATCH SCORING

For Match Scoring task, we adopt BERT-based and word2vec with WMD methods for comparing them with our proposed approaches described in Section III-C.

1) BERT-LARGE (FINE-TUNED)

BERT [12] is a language model which predicts word token occurrence probability from context before and after the token. BERT is based on transformer [48] and is trained on large amount of text data during the process of predicting automatically masked language or predicting next sentence from target sentence. The BERT algorithm consists of 12 or 24 transformer layers (called BERT-base and BERT-large, respectively). One of the main usages of BERT is fine-tuning. Fine-tuning is training on other dataset with setting the same initial parameters as in BERT.

Bar-Haim et al. experimentally showed that the fine-tuned BERT-large model yielded best scores when compared with other supervised methods [14]. It must be noted that their approach, unlike other methods described in this thesis, is supervised.

2) BERT-LARGE EMBEDDING

For comparison with other unsupervised approaches, we choose BERT-large [12] as the state of the art (SOTA) system in the existing unsupervised approaches for Match Scoring [14]. When we embed an argument and a key point using BERT-large, we examine averaged word embeddings as sentence embeddings. If the cosine similarity between an argument and a key point is above a threshold decided with the training data, the BERT-large method regards the pair as relevant.

3) Word2vec WITHOUT WMD

When we embed an argument and a key point using word2vec, we examine averaged word embeddings as sentence embeddings. If the cosine similarity between an argument and a key point exceeds a threshold decided with the training data, again the word2vec without WMD method regards the pair as relevant.

4) Word2vec WITH WMD

We experiment with Word Mover's Distance (WMD) [17] to compare with MoverScore-based method. WMD computes a distance between two texts using all distances between pairs of words with a embedding model (word2vec). This algorithm regards the distance between two texts as the minimum cost when it maps words in one text to words in another text. The cost between two words is calculated with cosine similarity using word2vec.

We compute distance between an argument and a key point using WMD. If the dissimilarity between an argument and a key point is below a threshold decided with the training data, the word2vec with WMD method regards the pair as relevant.

C. PROPOSED METHODS FOR MATCH SCORING

This section describes our proposed methods for Match Scoring task, i.e. Sentence-BERT, Sentence-BERT fine-tuned

The word2vec model we use is available at https://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negative300.bin.gz.
on STS benchmarks, which is explained in Section III-C2, and MoverScore. All of our proposed methods are using threshold-based methods like a comparison method, i.e., BERT-large Embedding, and word2vec approaches described in Sections III-B2, III-B3 and III-B4. Our methods decide that an argument matches a key point if the cosine similarity is above an automatically calculated threshold. If the similarity is lower than a threshold, it is decided that an argument does not match a key point. These thresholds for similarity function are acquired from the training data for each method. The detailed way to decide thresholds is as follows:

1) Calculating F1 score for the positive (matching) class for each threshold by classifying the training data;
2) Obtaining the thresholds which maximize the F1 score for the training data.

As a result, the thresholds of Sentence-BERT-base, Sentence-BERT-large, Sentence-BERT-STSb-base, Sentence-BERT-STSb-large, and MoverScore are 0.64, 0.71, 0.58, 0.56, and 0.12, respectively.

1) SBERT
Natural Language Inference (NLI) is a task in NLP which utilizes pairs of “hypothesis” and “premise” sentences. The task identifies whether these sentences are consistent (entailment), inconsistent (contradiction), or undetermined (neutral). We can say that the purpose of NLI task is to determine whether these sentences are similar to each other or not. This purpose resembles that of Match Scoring task, hence, we utilize Sentence-BERT model which is pre-trained on STS dataset [15]. This model is based on Siamese or triplet neural networks [28], [29].

Siamese networks consists of two and triplet networks of three networks trained with the same weights. These weights are trained on two different input vectors. Reimers and Gurevych [15] proposed Siamese BERT network and implemented it as Sentence-BERT6 (SBERT for short), which is trained on the following NLI datasets: Stanford Natural Language Inference (SNLI)3 [49] and Multi-Genre Natural Language Inference (MultiNLI)4 [50] in order to create universal sentence embeddings. SNLI is a dataset of 570,000 sentence pairs annotated with contradiction, entailment, and neutral labels, and the structure of MultiNLI is based on SNLI, and this corpus contains additional 430,000 sentence pairs and includes both spoken and written text.

For the following original Sentence-BERT configuration, we use the mean-pooling model.

2) SBERT-STSb
SBERT-STSb model is also proposed by Reimers and Gurevych [15]. This model is first fine-tuned on the AllNLI dataset (consisting both SNLI and MultiNLI datasets), then on the training set of STS benchmark [51].

STS benchmark9 stands for Semantic Textual Similarity, and this dataset includes English sentence pairs with the labels indicating how much two sentences are similar to each other. The label is evaluated with the continuous value from 0 to 5.

SBERT-STSb is well suited for measuring semantic textual similarity because originally this fine-tuning dataset is meant for text similarities.

3) MoverScore
MoverScore [16] is a robust evaluation metric for evaluating summarization or machine translation tasks. We adopt this scoring method to calculate similarity between an argument and a key point by measuring distances between words in each sentence. MoverScore is based on WMD or Sentence Mover’s Distance (SMD) [52]. To measure semantic distance, n-gram (n=1, 2) is utilized in addition to word distances. Word2Vec, ELMo, and BERT-base are adopted as embedding models for calculating word distances in MoverScore method. BERT-base is fine-tuned on following datasets: MultiNLI, Question Answering Natural Language Inference (QANLI)10, or Quora Question Pairs (QQP)11 [54], separately. Furthermore, the word representations from ELMo and BERT-base are aggregated with the following vector aggregation algorithms:

• Power Means [55]: an algorithm averaging a set of values with exponent function;
• Routing Mechanism [16]: a routing algorithm with Kernel Density Estimation.

A detailed explanation of Power Means algorithm which we use in Match Scoring task is described in Appendix A. ELMo and BERT output different vectors from each layer, therefore they aggregate vectors (all three layers for ELMo, and the final five layers for BERT-base).

We use the best metrics for summarization tasks reported in the MoverScore study [16] toward the aggregation of word representations, i.e., uni-gram, BERT-base, MultiNLI, Power Means, and WMD, to evaluate our proposed method. The MoverScore algorithm we utilized is implemented as follows:

1) BERT-base is fine-tuned on MultiNLI.
2) Each word in two arguments is embedded into five vectors with BERT-base model. These five embedding vectors originates from the last five layers.

9STS benchmark is available at https://github.com/facebookresearch/SemEval.
10QANLI dataset includes over 500,000 NLI data, which is automatically derived from a question answering dataset. QANLI corpus is available at https://worksheets.codalab.org/worksheets/0xd4ebc52cebb84130a07cbfe81597aaf0/.
11QQP is the largest dataset in GLUE benchmark [54], and it is a binary classification task whose goal is to determine whether two questions are semantically equivalent or not. QQP dataset is available at https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs.
Those five vectors are aggregated into one vector for each word using Power Means. Rare words are weighted using IDF value. WMD scores between one argument and another argument are computed, and the similarities are calculated according to the dissimilarities of WMD.

Note that the outputted score from the MoverScore metric ranges from $-1$ to $1$, and the higher score the more two texts are similar to each other. The lower score represents two texts are not similar.

To ensure the evaluation consistency, we followed the threshold decision process in the work of Bar-Haim et al. [14], and utilized F1 scores as a function for identifying optimal threshold in every method. All threshold decision curves in the proposed methods are shown in Figures 2 and 3.

### FIGURE 2
Threshold decision curves for the proposed methods. Blue, green, red, and black lines refer to Sentence-BERT-base, Sentence-BERT-large, Sentence-BERT-STSb-base, and Sentence-BERT-STSb-large, respectively.

### FIGURE 3
Threshold decision curve for the MoverScore-based approach.

### D. EMPIRICAL EVALUATION FOR MATCH SCORING
In this subsection, we describe our experiments with the baseline and proposed methods in order to measure the performance of our proposed approach. In the first subsection, we briefly describe the dataset which we used for the *Match Scoring* task. The second subsection lists the evaluation metrics for evaluating the baseline and proposed methods, and the last subsection shows the results of the evaluation experiments.

#### 1) EMPIRICAL EVALUATION DATA FOR MATCH SCORING
We utilize the dataset we describe in Section IV-A2. In this dataset, arguments which represent the same stance for each debate topic are grouped. The size of the dataset is 56 (28 debate topics for each stance).

#### 2) EVALUATION METRICS FOR MATCH SCORING
This section describes the evaluation metrics to evaluate the proposed and the baseline methods. Following Bar-Haim et al. [14], four evaluation metrics are utilized for the *Match Scoring* task: accuracy, precision, recall, and F1. These metrics are calculated with (1):

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN},
\]

\[
\text{Precision} = \frac{TP}{TP + FP},
\]

\[
\text{Recall} = \frac{TP}{TP + FN},
\]

\[
F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}},
\]

where TP refers to True Positive, TN stands for True Negative, FP means False Positive, and FN refers to False Negative.

### E. EVALUATION RESULTS FOR MATCH SCORING
The results of accuracy, precision, recall, and F1 score on the test dataset are shown in Table 3-E. These are results of experiments on the test data using thresholds learned from the training data. As reported in Table 3-E, SBERT-large achieves higher scores in accuracy, precision, and recall than an existing unsupervised SOTA method, i.e. BERT Embedding. While SBERT-large achieves a relatively good score, versions of SBERT-STSb do not exceed SBERT-large even

| Method                | Acc  | P    | R    | F1   |
|-----------------------|------|------|------|------|
| SBERT-large           | 0.673| 0.290| 0.501| 0.368|
| SBERT-large            | 0.493| 0.218| 0.635| 0.324|
| SBERT-STSb-base       | 0.682| 0.317| 0.571| 0.408|
| SBERT-STSb-large      | 0.726| 0.354| 0.513| 0.419|
| MoverScore            | 0.706| 0.335| 0.544| 0.415|
| MoverScore            | 0.747| 0.388| 0.554| 0.457|

where TP refers to True Positive, TN stands for True Negative, FP means False Positive, and FN refers to False Negative.
though they have been reported as having superior capability for measuring semantic similarity [15]. For this reason, before the experiment, we expected that SBERT-STSb would exceed SBERT score. STS benchmark and other NLI datasets contain daily conversation-like data, and our intuition is that such data might be unsuitable for debate topics. In future, debate or argument-oriented BERT model (like BERT trained on Google News or BERT trained on debate data) could be used instead of the standard BERT [23].

Moreover, MoverScore yields the best scores except the recall. MoverScore relies on soft-alignment (many-to-one) for input words, thus it allows to map each single word in one text to their semantically related words in another text. This is, in our opinion, why the MoverScore performed best.

Regarding the recall, word2vec with WMD yields the best results. However, its accuracy, precision and F1 values are the worst when compared with other methods. Precision and recall are calculated with (1).

As can be observed in the results, in case of word2vec with WMD, many false positives are discovered. From the perspective of precision, it can be said that the threshold of word2vec with WMD does not have sufficient capability to identify positive and negative samples. From the precision results, it can be concluded that the small number of false negative causes an increase in recall in the case of word2vec with WMD.

Regarding the recall, it seems that it cannot evaluate approaches properly. This problem seems to originate in the data bias. As we have already mentioned in Section III-B, more non-matching pairs exist than matching ones. The number of matching pairs (positive) is 4,998 and the number of non-matching pairs (negative) is 19,095. Therefore, if a method does not have sufficient prediction ability, it may tend to label pairs as negative because the number of negative samples is incomparably larger. Therefore, in the case of ArgKP dataset, it is better to focus on precision rather than recall.

Furthermore, the above-mentioned recall problem is also the reason why the recall in word2vec with WMD is higher than that in MoverScore.

F. ERROR ANALYSIS FOR MATCH SCORING

In this subsection, we present error analysis for the Match Scoring task.

Within the topic “We should ban the use of child actors,” argument “the use of child actors exploits a child and can have negative effects on them” and key point “Being a performer harms the child’s education” represent different content. However, the similarity between the argument and the key point is high (0.914). The reason for this may be that both argument and key point describe negative effects on child actors, so our MoverScore-based method incorrectly classified them as representing the same content. Both argue the negative effects on children, but from different facets, such as “the use of child actors exploits a child” and “Being a performer harms the child’s education”. It can be assumed that our model will obtain better results if it can identify the difference from a perspective of detecting different facets of a discussed problem.

This problem occurs not only in both of our methods, but also in SBERT and MoverScore. We plan to alleviate it in near future methods utilized in the Argument Facet Similarity task [13], which is described in Section II-A.

G. DISCUSSION ON MATCH SCORING

In this section, we discuss the results of the Match Scoring task and explain how to utilize this task results for Argument Extraction.

The existing methods, which average word embeddings to compute sentence embeddings, cannot weight important words in an argument, therefore Sentence-BERT model and MoverScore metric are utilized for the task. Both of the proposed approaches outperformed the state of the art system (BERT-large Embedding) on the ArgKP dataset [14]. The results suggest that our methods successfully weight important words which was not achieved in other studies.

From these results, it can be concluded that the sentence embedding models and NLI dataset, which is used for fine-tuning and training Sentence-BERT and MoverScore metric, are useful for the Match Scoring task, so we plan to utilize other sentence transformers, like Sentence-RoBERTa which is also fine-tuned on NLI dataset, to see if it is able to improve our results.

MoverScore we used for the proposed method is based on Word Mover’s Distance, which means that both word significance and word meaning are represented by the same vectors. These two properties for words should be considered individually. Therefore, we plan to adopt Word Rotator’s Distance, which can represent the importance and the meaning separately.

From these results, we can say that MoverScore metric is useful for computing argument similarity. Our proposed method for Argument Extraction described in Section IV-C utilizes argument similarity, therefore MoverScore is adopted to measure similarity between two arguments. Furthermore, in the next section, we compare MoverScore-based method with Sentence-BERT-large-based one, which is the second best approach in Match Scoring task.

IV. ARGUMENT EXTRACTION

In this Section, we describe the Argument Extraction task. As the first step, we extract key points from a set of arguments for each topic (divided into pro and con). We explain how we create our data for Argument Extraction from ArgKP dataset, and attempt to extract the accurate arguments from a set of all arguments using unsupervised methods in order to generate key points. Finally, we evaluate the output with two original evaluation metrics. The number of extracted arguments is changed from one to twenty, and this study analyzes the outputs and evaluate them with two newly proposed evaluation metrics.
A. EXPERIMENTAL SETUP FOR ARGUMENT EXTRACTION

In this subsection, our preparations for Argument Extraction task are described: the task setting, the data for the task, and its evaluation metrics.

1) TASK EXPLANATION OF ARGUMENT EXTRACTION

In Argument Extraction, we extract some arguments from an argument set and generate key points for each topic (arguments are divided into each stance, i.e. pro and con). Figure 4 introduces a case where extracted arguments are perfectly matched with related key points. In this case, we assume that there are 10 arguments in a debate topic for the pro side. Proposed approaches and baseline methods for the task try to extract an adequate argument for each key point. For instance, in Figure 4, if the Argument 1 (Arg) is extracted, it can be said that Key Point 1 (KP) is represented by a set of extracted arguments (in this paper we describe them as key points being “covered” by arguments). Therefore, in this example, a system needs to extract each adequate argument for two key points, such as the Arguments 1 and 7.

As shown in Figure 4, there exist some arguments which do not belong to any key point, i.e. arguments 9 and 10. These arguments are treated as noise in this task.

2) DATA FOR ARGUMENT EXTRACTION

We utilize ArgKP dataset for Argument Extraction for creating a set of various arguments. In order to adapt ArgKP to this task, arguments are grouped in the same debate topic and the same stance (pro and con). In total, the created dataset includes 56 sets (28 debate topics for each stance). Each set contains 122 arguments, and 4.34 key points on average.

TABLE 4. Frequency of samples by the number of key points in a set of data.

| Number of Key Points | 1   | 2   | 3   | 4   | 5   | 6   | 7   |
|----------------------|-----|-----|-----|-----|-----|-----|-----|
| Frequency            | 1   | 4   | 17  | 20  | 8   | 0   |     |

In Section I, it is reported that approximately 72.5% of arguments can be gathered with seven key points. In the ArgKP dataset creation process, key points with low annotator agreement (Cohen’s kappa lower than 0.82) were deleted [14]. Frequencies of key points included in each stance in each debate topic are shown in Table 4. It is revealed that no set with seven or more key points exists in the whole dataset.

TABLE 5. Average number of arguments in a set under the corresponding number of key points.

| # of Key Points | 1   | 2   | 3   | 4   | 5   | 6   | 7   |
|-----------------|-----|-----|-----|-----|-----|-----|-----|
| # of Arguments  | 80.0| 102.8| 112.5| 116.8| 119.3| 122.3|     |

Each set consists of approximately a hundred of arguments. Table 5 shows the average number of arguments in a set under the corresponding number of key points. It can be observed that the number of arguments gradually increases along with the number of key points, but the differences between the averages are not large.

In our dataset, there is also one set which includes only one key point. Its debate topic is “We should fight for the abolition of nuclear weapons”, and the key point is “Nuclear weapons is [sic] essential for protection and deterrence” as shown in Table 6. In this set, arguments represent the con
A link is created between two words if one follows another, and link weight is calculated and link weight increases if these two words occur more frequently adjacent to each other. The link weight is calculated by finding words occurring close to each another.

Mihalcea and Tarau [35] proposed TextRank, a ranking algorithm based on PageRank [56], which is often used in keyword extraction and text summarization. In order to find relevant keywords, TextRank constructs a word / sentence network by finding words occurring close to each other. A link is created between two words if one follows another, and link weight increases if these two words occur more frequently adjacent to each other. The link weight is calculated as follows:

$$\text{Sim}(D_1, D_2) = \frac{n_{1\&2}}{\log(n_1) + \log(n_2)},$$

(2)

where $D_1$ and $D_2$ are sentences (nodes in the network), $n_1$ and $n_2$ are the numbers of words included in $D_1$ and $D_2$, and $n_{1\&2}$ the number of the words overlapping between $D_1$ and $D_2$.

We adopt TextRank as a baseline method to extract various arguments from a set of arguments. With TextRank, we extract the top key size arguments as key points. Note that key size is a number of arguments to extract, and key size is changed from 1 to 20 in order to explore which key size is appropriate to extract various arguments.

### B. BASELINE METHOD FOR ARGUMENT EXTRACTION

Mihalcea and Tarau [35] proposed TextRank, a ranking algorithm based on PageRank [56], which is often used in keyword extraction and text summarization. In order to find relevant keywords, TextRank constructs a word / sentence network by finding words occurring close to each other. A link is created between two words if one follows another, and link weight increases if these two words occur more frequently adjacent to each other. The link weight is calculated as follows:

$$\text{Sim}(D_1, D_2) = \frac{n_{1\&2}}{\log(n_1) + \log(n_2)},$$

(2)

where $D_1$ and $D_2$ are sentences (nodes in the network), $n_1$ and $n_2$ are the numbers of words included in $D_1$ and $D_2$, and $n_{1\&2}$ the number of the words overlapping between $D_1$ and $D_2$.

We adopt TextRank as a baseline method to extract various arguments from a set of arguments. With TextRank, we extract the top key size arguments as key points. Note that key size is a number of arguments to extract, and key size is changed from 1 to 20 in order to explore which key size is appropriate to extract various arguments.

### C. PROPOSED METHOD FOR ARGUMENT EXTRACTION

In this section, we introduce our proposed method for Argument Extraction. First, we explain Maximal Marginal Relevance (MMR) which is adopted in order to extract various arguments from a set of arguments. Secondly, we describe similarity function used in MMR algorithm. Finally, we clarify how we use MMR for the Argument Extraction task.

1) **MMR ALGORITHM**

This section describes Maximal Marginal Relevance (MMR) algorithm [32], which we use for extracting arguments. MMR algorithm is usually utilized for key word extraction without reference sentences [37]. We adapt it in order to extract arguments from a set of arguments.

MMR algorithm controls the diversity of extracted key phrases and the relevance between documents and extracted phrases. It uses two parameters: key size and $\lambda$.

Key size defines the number of extracted key phrases. When key size equals 1, only one key phrase is extracted by MMR algorithm, and when key size equals 5, five key phrases are chosen. $\lambda$ is a parameter introduced in (3) and its role is to specify the trade off between relevance and redundancy of the MMR algorithm. With a given input query $Q$, the set $S$ represents documents that are selected as correct answers for $Q$. $S$ is populated by computing MMR score as described in (3):

$$\text{MMR} := \arg \max \{ \lambda \cdot \text{Sim}_1(D_i, Q) \}
- (1 - \lambda) \max \{ \text{Sim}_2(D_i, D_j) \}$$

(3)

where $R$ is the ranked list of documents retrieved by an algorithm, $S$ represents the subset of documents in $R$ which are already selected, $D_i$ and $D_j$ are retrieved documents, and $\text{Sim}_1$ and $\text{Sim}_2$ are similarity functions. The smaller $\lambda$ becomes, the diversity of outputs increases.

2) **SIMILARITY FUNCTIONS FOR MMR ALGORITHM**

We propose separate similarity functions for $\text{Sim}_1$ and $\text{Sim}_2$ introduced in (3), i.e. MoverScore-based similarity.

From the results described in Chapter III, it can be said that MoverScore algorithm (see Section III-C2), is more useful for computing semantic distances between arguments than other unsupervised methods including Sentence-BERT-based approaches. Therefore, in order to compute similarity used in (3), we additionally adopt MoverScore-based similarity. As described in Chapter III, the MoverScore we use is based on unigrams, BERT-base, MNLI, Power Means, and Word Mover’s Distance. There are two similarity functions in MMR: the first one is similarity between a whole document and one input phrase, and the second one is similarity between one input phrase and one phrase which has been extracted beforehand. The second similarity function $\text{Sim}_2$ can be modified with MoverScore algorithm as follows:

$$\text{Sim}_2(D_i, D_j) = \text{MoverScore}(D_i, D_j).$$

(4)

There are no documents in our dataset, therefore, in order to compute the first similarity function $\text{Sim}_1$, we compute MoverScores between all sentence pairs. Hence, we calculate

### TABLE 6. An example set of a debate topic and its stance, which includes only one key point. Argument Cover Rate is explained in Section IV-D2.

| Debate Topic: We should fight for the abolition of nuclear weapons | Stance: Con |
|---|---|
| The Number of Arguments: 80 | |
| Key Point: Nuclear weapons is essential for protection and deterrence | |
| Argument Cover Rate: 93.75% | |
the similarity function $\text{Sim}_1$ with (5) as follows:

$$\text{Sim}_1(D_i, D) = \sum_{k=1}^{N} \text{MoverScore}(D_i, D_k). \tag{5}$$

where $D$ is the whole set of documents, and $N$ is the number of documents included in $D$.

3) COMPARISON OF SIMILARITY FUNCTIONS

Next, we describe our comparison experiment for the similarity function in (3) in order to examine if our proposed algorithm achieves a superior performance when compared to other methods. To compare with MoverScore-based similarity function, we experiment with the cosine similarity using Sentence-BERT.

From the results described in Chapter III, it can be said that Sentence-BERT-large,\footnote{We use Sentence-BERT model available at https://github.com/UKPLab/sentence-transformers, which is the same model to the one used in Chapter III.} which is explained in Section III-C1, is more useful for computing sentence embeddings of arguments than other Sentence-BERT-based methods. Therefore, in order to compare Sentence-BERT-based method with our proposed one, we also experiment with Sentence-BERT-based cosine similarity. This similarity function is calculated with (6):

$$\text{Sim}(D_i, D_j) = \cos(\text{func}(D_i), \text{func}(D_j)) \tag{6}$$

where $D_i$ and $D_j$ are input documents, $\cos(v1, v2)$ is a cosine similarity function between two vectors $v1$ and $v2$, and $\text{func}(D)$ is an sentence embedding model of a document $D$. As mentioned before, we utilize Sentence-BERT for the sentence embedding model.

Both similarity functions used in MMR score originate from (6).

4) MMR ALGORITHM IN OUR PROPOSED METHOD

Maximal Marginal Relevance (MMR) is an algorithm for selecting various words from a text, and the system performs better for choosing words which are not similar to already extracted ones. Therefore, we adopt it in order to extract diverse arguments from a set of arguments.

In (3), the first term $\text{Sim}(\text{Arg}_i, Q)$ represents the similarity between a given argument $\text{Arg}_i$ and all arguments in a set $Q$. The second term $\max_{\text{Arg}_j \in K} \text{Sim}(\text{Arg}_i, \text{Arg}_j)$ represents the difference between a given argument $\text{Arg}_i$ and an already extracted argument. This second term makes MMR score lower if a given argument is similar to already selected arguments, hence, MMR score algorithm would extract diverse arguments.

In the proposed method, both similarity functions are computed with (5) and (4), respectively. $\lambda$ parameter in MMR algorithm is set to 0.5 following the default value reported in MMR algorithm.

Using both similarity functions and $\lambda$ parameter, we score each argument and extract the top key size arguments via (3).

Note that key size is a number of arguments to extract, and key size is changed from 1 to 20 as well following the baseline method.

D. EMPIRICAL EVALUATION FOR ARGUMENT EXTRACTION

In this section, we describe the experiments with the baseline and proposed methods in order to measure the usefulness of our approach. In the first subsection, we briefly describe the dataset which we used for the Argument Extraction task. The second subsection lists our proposed evaluation metrics for evaluating the baseline and proposed methods, and the last subsection shows the results.

1) DATA FOR EMPIRICAL EVALUATION IN ARGUMENT EXTRACTION

We utilize the dataset described in Section IV-A2. In this dataset, arguments which represent the same stance for each

| Debate Topic: We should end mandatory retirement | The Number of Arguments: 78 |
|-------------------------------------------------|----------------------------|
| The Stance: Con                                 |                            |
| KP$_1$ (71%) A mandatory retirement age creates opportunities for other workers | KP$_2$ (4%) mandatory retirement age benefits those that are about to be retired |

| Debate Topic: We should fight for the abolition of nuclear weapons | The Number of Arguments: 116 |
|--------------------------------------------------------------------|----------------------------|
| The Stance: Pro                                                   |                            |
| KP$_1$ (10%) Nuclear weapons can fall into the wrong hands        | KP$_2$ (9%) Nuclear weapons provoke or escalate a disastrous military action |

| Debate Topic: We should prohibit flag burning                     | The Number of Arguments: 99 |
|--------------------------------------------------------------------|----------------------------|
| The Stance: Pro                                                   |                            |
| KP$_1$ (40%) flag burning is an inflammatory act                   | KP$_2$ (58%) Flags are important symbols that should be respected |

| Debate Topic: We should prohibit flag burning                     | The Number of Arguments: 118 |
|--------------------------------------------------------------------|----------------------------|
| The Stance: Con                                                   |                            |
| KP$_1$ (81%) flag burning is justified due to freedom of expression | KP$_2$ (16%) flag burning is an effective tool of expression/protest/critique |
debate topic are grouped. The size of the dataset is 56 (28 debate topics for each stance).

2) EVALUATION METRICS FOR ARGUMENT EXTRACTION
Below we introduce two evaluation metrics for the Argument Extraction task: Key Point (KP) Cover Rate, and Argument Cover Rate.

Note that, for the reasons explained in Section IV-A2, the proposed metrics do not consider how many key points an extracted argument corresponds to.

3) KEY POINT (KP) COVER RATE
We propose a metric for evaluation which relies on measuring how many key points all extracted arguments belong to. In our dataset, each argument is tagged with its associated key points, so it can be easily revealed if extracted arguments correspond to key points in a given argument set. Therefore, we divide the number of these corresponded key points by the number of all key points in a given set and define the result as Key Point (KP) Cover Rate. Hence, the KP Cover Rate is represented as follows (7);

\[
CoverRate(KP) = \frac{ExtractedKPs}{AllKPs},
\]

where \( ExtractedKPs \) is the number of the key points which extracted arguments belong to, and \( AllKPs \) is the number of all key points included in a given set.

4) ARGUMENT COVER RATE
We propose a metric for evaluation which measures how many arguments from all extracted arguments are similar. In our dataset, each argument is tagged with its associated key points, thus it is obvious which key points are associated with the extracted arguments. These associated key points are also associated with other arguments in the same group. Therefore, it is clear which associated arguments are related to the extracted arguments.

For example, in Figure 4, if an Argument 1 is extracted, it belongs to the Key Point 1. The Key Point 1 corresponds to arguments 1, 2, 3, 4, and 5. It can be said that these arguments are associated with the extracted argument (Argument 1) because their tagged key point is identical.

Based on this idea, we evaluate how many arguments all extracted arguments are similar to. We divide the number of the associated arguments with the number of all arguments in a given set. Then, we define the result as Argument Cover Rate.

Argument Cover Rate is represented with (8);

\[
CoverRate(Argument) = \frac{ExtractedArgs}{AllArgs},
\]

where \( ExtractedArgs \) is the number of the arguments which belong to key points that extracted arguments correspond to, and \( AllArgs \) is the number of all arguments included in a given set.

5) EVALUATION RESULTS IN ARGUMENT EXTRACTION
In this section, we describe KP Cover Rate and Argument Cover Rate evaluation results for extracted arguments.

Figures 5 and 6 show comparison of both proposed evaluation metrics for the Argument Extraction methods evaluated on all data. Note that in the experiments described in this Section we do not divide the dataset, therefore all data is used for testing.

Blue, orange, and green lines refer to TextRank, Sentence-BERT-based MMR method, and our proposed method (i.e. MoverScore-based MMR method), respectively. The detailed results of Argument and Key Point (KP) Cover Rates are listed in Tables 9 and 8.

Regarding the KP Cover Rate, for all key sizes, the MoverScore-based MMR method yields the best scores. This indicates that MoverScore we used to calculate argument similarity is beneficial as we expected from the experimental results described in Section III.

The difference between the Sentence-BERT-based MMR method and TextRank is not significantly large, and from
TABLE 9. Detailed results of argument cover rates [%].

| key size | TextRank | SBERT-based MMR | MoverScore-based MMR |
|----------|----------|------------------|----------------------|
| 1        | 20.0     | 44.0             | 26.3                 |
| 2        | 36.9     | 35.2             | 39.3                 |
| 3        | 42.4     | 39.3             | 50.2                 |
| 4        | 44.4     | 44.0             | 58.0                 |
| 5        | 48.0     | 48.3             | 59.5                 |
| 6        | 50.5     | 51.2             | 62.9                 |
| 7        | 52.5     | 54.4             | 64.0                 |
| 8        | 54.0     | 56.1             | 65.3                 |
| 9        | 55.0     | 58.2             | 66.1                 |
| 10       | 57.9     | 58.9             | 67.5                 |
| 11       | 58.8     | 60.2             | 68.7                 |
| 12       | 59.9     | 61.2             | 69.3                 |
| 13       | 60.4     | 62.8             | 69.5                 |
| 14       | 60.8     | 63.5             | 71.0                 |
| 15       | 61.2     | 64.0             | 71.5                 |
| 16       | 61.4     | 64.3             | 72.3                 |
| 17       | 61.8     | 65.0             | 72.4                 |
| 18       | 61.9     | 65.7             | 72.6                 |
| 19       | 62.4     | 65.8             | 73.2                 |
| 20       | 62.7     | 66.0             | 73.2                 |

As mentioned in Section III, the proposed MoverScore-based method is the best unsupervised algorithm to compute argument similarity between arguments and key points in our experimental setup. Accordingly, the proposed MoverScore-based MMR method yields the best in both metrics. In both metrics, it can be said that MoverScore-based MMR approach performs best when all key sizes are considered, but when key sizes is small, especially key size = 1 or 2, the differences in results are not large.

E. ERROR ANALYSIS IN ARGUMENT EXTRACTION

In this section, we analyze errors concentrating on two examples: when key size = 1 and key size = 8.

1) ERRORS OCCURRING WHEN KEY SIZE EQUALS 1

The third column of Table 10 indicates how many key points are covered by extracted arguments using each method when the key size is equal to 1. When TextRank method is used, there are three cases of covering two key points. On the contrary, each of MMR-based methods has only one such case. Our metrics can properly evaluate these cases, but as mentioned in Section IV-A2, extracting arguments associating two or more key points is not desirable for reasons explained in Section IV-A2.

2) ERROR EXAMPLES WHEN KEY SIZE EQUALS 8

There are some cases when the MoverScore-based MMR approach does not work well. For example, in a debate topic “We should abandon marriage”, for the pro stance,

key size = 2 to key size = 4, TextRank outperforms the Sentence-BERT-based MMR approach.

Regarding the Argument Cover Rate, identically as in the case of key points, for all key sizes, the MoverScore-based MMR method yields the best scores. The difference between the Sentence-BERT-based MMR method and TextRank is not significant when key size is small.

Both cover rates gradually increase for all methods as key size decreases. This is natural because the number of covered arguments widens their topical variety along with the increase of the key size. However, the difference between TextRank and the MMR-based methods is larger when compared with the KP Cover Rate. The experimental results show that TextRank is not able to select various types of arguments, while MMR can extract them as we hypothesized in Section IV-C.
TABLE 11. Example arguments extracted by MoverScore-based MMR method when the key size is 8 and the list of corresponding key points for each argument. KP refers to a key point, Arg refers to argument, and the i number next to Arg, refers to the i-th argument extracted by MMR score, meaning that Arg_1 is the highest scored argument.

| Debate Topic: We should abandon marriage | Stance: Pro | The Number of Arguments: 125 | The Number of Key Points: 4 |
|-----------------------------------------|-------------|-------------------------------|-----------------------------|
| KP_1  | Marriage preserves norms that are harmful for women | Covered KP by MoverScore-based MMR: KP_1, KP_2, KP_3, KP_4 | Covered Argument: Arg_1-Arg_8 |
| KP_2  | Marriages are unstable | Covered Argument: Arg_1-Arg_8 |
| KP_3  | Most of the advantages of marriage are achieved by a monogamous relationship | Covered Argument: Arg_1-Arg_8 |
| KP_4  | Marriages tie up people with unfair obligations | Covered Argument: Arg_1-Arg_8 |

| Covered KP by Sentence-BERT-based MMR | Covered KP by MoverScore-based MMR |
|--------------------------------------|-----------------------------------|
| KP_1  | Covered Argument: Arg_1-Arg_8 |
| KP_2  | Covered Argument: Arg_1-Arg_8 |
| KP_3  | Covered Argument: Arg_1-Arg_8 |
| KP_4  | Covered Argument: Arg_1-Arg_8 |

ID | Arguments | Covered KP by Sentence-BERT-based MMR | Covered KP by MoverScore-based MMR |
---|-----------|-------------------------------------|----------------------------------|
Arg_1 | marriage is an outdated social construct | None | None |
Arg_2 | marriage is an outdated concept which allowed fathers to sell their daughters and is still used in cultures to force women to marry their rapists | None | None |
Arg_3 | marriage is an archaic practice that was originally formed to work as a business arrangement rather than for love | None | None |
Arg_4 | we should abandon marriage because people don’t stay married too often nowadays anyway | KP_2 | KP_2 |
Arg_5 | marriage is just a government scam to tax you for your love | None | None |
Arg_6 | we should abandon marriage because a lot of people don’t take their wedding vows seriously anyway and cheat on their spouse | None | None |
Arg_7 | too many marriages end in divorce, or worse; civil unions offer almost the same benefits as marriage anyway | None | None |
Arg_8 | marriage comes with an unrealistic expectation that partners be compatible for life | None | None |

F. DISCUSSION ON ARGUMENT EXTRACTION

In this section, the discussion on several experimental results is described.

1) CASE WHEN KEY SIZE IS SMALL

Here we discuss the case when key size is small (e.g., key size = 1 or key size = 2).

As mentioned in Section IV-D5, when key size is small, especially key size = 1 or 2, the differences in both metrics between the proposed method and TextRank approach are not too significant. There are two possible reasons for this.

Firstly, the baseline method tends to extract an argument which belongs to two or more key points in the first extracting step. For example, Table 12 shows an example of key points to which TextRank, Sentence-BERT-based MMR method and MoverScore-based method extracted following arguments:

- **TextRank**: “compulsory voting makes the population more involved in the election process, and would give results that more accurately reflected the will of the electors.”

- **Sentence-BERT-based MMR**: “we should introduce compulsory voting since everyone should vote and what better way to encourage voting than by making mandatory.”

In this case, MMR scoring algorithm works well for the MoverScore-based MMR method, because all selected arguments differ from each other, and the key point which the Arg_1 belongs to is not covered by other extracted arguments. However, only single key points is covered. In our proposed method, the central vector is computed from all argument embeddings, so dividing arguments and calculating several central vectors would be useful in order to map arguments into several clusters more precisely. Introducing a clustering step before calculating MMR score could be beneficial for the argument grouping step. First, one would need to cluster arguments into several groups, and then to compute similarities between the cluster centers and input arguments. Secondly, one could utilize these similarities replacing Sim(Arg_i, Q) in (3).

We assume that clustering step could capture common argument groups, i.e., key points, therefore we assume that introducing clustering might be useful for solving the problem of extracting arguments not belonging to any key point.

TABLE 12. Example of key points covered by an extracted argument when key size = 1.

| Debate Topic: We should introduce compulsory voting | Stance: Pro | The Number of Arguments: 115 |
|------------------------------------------------------|-------------|----------------------------|
| Key Point List | Compulsory voting increases representation | People have an obligation to vote | Compulsory voting will cause more people to become interested in politics |
| TextRank | Covered | Not Covered | Covered |
| SBERT-based | Not Covered | Covered | Not Covered |
| MoverScore-based | Covered | Not Covered | Not Covered |
MoverScore-based MMR: “compulsory voting would ensure that the person or party elected is the one that genuinely has the most support of the entire electorate.”

As shown in Table 12, an extracted argument by the TextRank approach belongs to two out of three key points. This causes that the cover rate of key points becomes larger with the TextRank method when the key size is small.

The second reason is that MMR score algorithm does not work well when the number of extracted arguments is small. As described in Section IV-C, this algorithm puts more weight on arguments which are different from the set of already extracted arguments. However, if the number of already extracted arguments is zero (when key size = 1) or small, diverse arguments cannot be selected, thus the MMR algorithm performs similarly to TextRank when key size is small.

These two reasons are the possible causes why there is no difference between the proposed method and TextRank when the key size is not large.

2) COMPARISON BETWEEN HUMAN-MADE KEY POINTS AND ARGUMENTS EXTRACTED WITH OUR PROPOSED METHODS

In this subsection, we compare the original key points with the arguments extracted with our proposed method. Furthermore, we also analyze the original arguments with ones selected by the methods used for comparison.

The MoverScore-based MMR method obtains the best results in two argument similarity calculation metrics for all key sizes. As an example of the extracted arguments using this proposed approach, Table 13 lists extracted con stance arguments in a debate topic “We should abolish capital punishment”. Arg and KP in this table refer to extracted arguments and key points. The number i next to Arg_i refers to the i-th extracted by MMR score. Each argument in the list, except Arg_2, corresponds to only one key point, which is a desirable result.

In this case, all key points are covered when key size = 8 as shown in Table 13. The table also shows which argument is associated with which key point.

It is necessary to compare extracted arguments by the proposed methods with the original key points in order to explore which key size performs similarly to human experts. As mentioned in Section II, Bar-Haim et al. [14] reported that key points cover approximately 72.5% of arguments. As shown in Figure 6, it is clear that the proposed method, i.e. MoverScore-based MMR approach, can cover as many arguments as in the experts when key size = 11 (72.5%).

With regards to other methods, the TextRank algorithm did not exceed 72.5% until key size is equal to 20, and the Sentence-BERT-based MMR approach can cover over 72.5% arguments from key size = 17 (73.4%). From these results, it can be concluded that key size = 11 is enough for covering arguments.

3) FUTURE WORK FOR ARGUMENT EXTRACTION TASK

In near future, as mentioned in Section IV-E, we plan to introduce the argument length penalty into MMR score in order to suppress arguments belonging to two or more key points.

Furthermore, clustering step is also most likely needed when calculating MMR score for recognizing key point groups, in order to solve the problem that arguments not associated to any key point are extracted.

V. CONCLUSION

In this paper, two tasks for easy access to arguments were introduced: Match Scoring and Argument Extraction. In order to retrieve core arguments (key points), we attempted to extract arguments which represent key points for each debate topic.

In the existing Match Scoring task, we focused on identifying whether arguments belong to key points which were

| Debate Topic | We should abolish capital punishment |
|--------------|--------------------------------------|
| Stance: Con  |                                      |
| The Number of Arguments: 110 |
| The Number of Key Points: 5 |
| KP1: The death penalty is proportional |
| KP2: The death penalty is acceptable in cases of severe crimes |
| KP3: The death penalty prevents crimes |
| KP4: The death penalty helps the victim/their family |
| KP5: The death penalty saves costs to the state |
| Covered KP by TextRank | KP1, KP2, KP3 |
| Covered KP by Sentence-BERT-based MMR | KP2, KP3, KP4 |

| ID | Arguments | Covered KP |
|----|-----------|------------|
| Arg1 | capital punishment is the only real solution for criminals who commit terrible crimes. | KP2 |
| Arg2 | the death penalty gives the victims of the worst crimes justice and takes away the option that they could be released to commit these crimes again | None |
| Arg3 | too much of my taxes go towards paying for bad people. capital punishment would reduce jail spending | KP3 |
| Arg4 | capital punishment provides solace to the loved ones of the deceased that the criminal will not be set free again. | KP4 |
| Arg5 | some murderers are so brutal they must be executed | KP2 |
| Arg6 | capital punishment must be retained as the ultimate punishment for the most heinous crime. an example needs to be shown to others that this in a consequence of your actions think again. | KP3 |
| Arg7 | the punishment should be equal to the crime. it also might deter others from murder if they know they will lose their own life. | KP3 |
| Arg8 | capital punishment provides a punishment that is equal to the crime committed. if you kill you should expect to be killed. | KP1 |

TABLE 13. Example of extracted arguments using the MoverScore-based MMR method when key size = 8 and the list of corresponding key points for each argument. KP refers to key point, Arg refers to argument, and the number i next to Arg_i refers to the i-th extracted by MMR score.
created by debate experts for each topics. We pointed out that the existing methods which average word embeddings to compute sentence embeddings, did not weight important words in an argument, therefore we proposed to utilize Sentence-BERT model and MoverScore metric for the task, and both of our proposed methods outperformed the state of the art system (BERT-large) on the ArgKP dataset [14], which includes [argument, key point] pairs for Match Scoring task. This could be due to the fact that our methods weight important words which was not performed in other studies. From the experimental results, it can be concluded that the sentence embedding models and NLI dataset, which was used for fine-tuning Sentence-BERT and MoverScore metric, were useful for the Match Scoring task. The experiments also showed that MoverScore-based approach performs best in all unsupervised methods, therefore we utilized MoverScore for our proposed method for the Argument Extraction task.

In Argument Extraction task, arguments which accurately corresponded to key points were selected from a set of all arguments. MMR approaches worked well to find out diverse arguments, and MoverScore-based MMR achieved the best results. However, although MoverScore-based MMR worked well in the most cases, the system was sometimes extracting arguments belonging to two or more key points, occasionally causing erroneous associations. To tackle this problem, we plan to add the argument length penalty function into MMR score function as one of possible solutions. Moreover, adding a clustering step might be helpful to identify arguments which correspond to key points because key points might be the center of clusters in the arguments.

In the future, if supervised learning methods are used in addition to unsupervised approaches we utilized, the size of the dataset also becomes a problem, hence data augmentation is one of the most crucial techniques for obtaining sufficient data for training.

**APPENDIX A**

**POWER MEANS FOR MoverScore**

In this section, Power Means used in the MoverScore algorithm is explained.

Power Means algorithm averages a set of values with exponent function as described in (9). In MoverScore it aggregates word representations \( (z_{i,l})_{l=1}^{L} \) computed by BERT. When we set \( p \in \mathbb{R} \cup \pm \infty \), Power Means algorithm computes the mean of \( z_{i,1}, \ldots, \) and \( z_{i,L} \) as shown below:

\[
h_{i}^{(p)} = \left( \frac{z_{i,1}^{(p)} + \cdots + z_{i,L}^{(p)}}{L} \right)^{\frac{1}{p}} \in \mathbb{R}^{d}. \tag{9}
\]

Power Means algorithm can set \( p \) to several values. In such cases, the system computes word embeddings by concatenating as represented in (10):

\[
h_{i}^{(p_1)} \oplus \cdots \oplus h_{i}^{(p_k)}, \tag{10}
\]

where \( K \) is a number of \( p \) which MoverScore metric uses, and \( \oplus \) is a vector concatenation. In MoverScore metric, \( K \) is set to 3, and \( p \) is set to 1 and \( \pm \infty \).

**APPENDIX B**

**DATA EXAMPLE FOR ARGUMENT EXTRACTION**

This appendix shows examples of data used for Argument Extraction task.

**A. EXAMPLE OF SETS INCLUDING THREE KEY POINTS**

Table 14 exemplifies a set which includes three key points. This table also shows how many arguments each key point covers.

| Debate Topic: We should adopt atheism | Stance: Con | The Number of Arguments: 123 |
|---------------------------------------|-------------|-------------------------------|
| Key Point 1: Atheism discriminates against religious people / violates freedom of religion |
| Cover Rate | 31.7% |
| Key Point 2: Atheism lacks morality |
| Cover Rate | 13.8% |
| Key Point 3: People should choose for themselves whether Atheism should be adopted |
| Cover Rate | 9.8% |

**B. EXAMPLE OF SETS INCLUDING FOUR KEY POINTS**

Table 15 exemplifies a set which includes four key points. This table also shows how many arguments each key point covers.

| Debate Topic: Assisted suicide should be a criminal offence | Stance: Con | The Number of Arguments: 121 |
|-----------------------------------------------------------|-------------|-------------------------------|
| Key Point 1: Assisted suicide gives dignity to the person that wants to commit it |
| Cover Rate | 11.6% |
| Key Point 2: The terminally ill would benefit from assisted suicide |
| Cover Rate | 8.3% |
| Key Point 3: Assisted suicide reduces suffering |
| Cover Rate | 51.2% |
| Key Point 4: People should have the freedom to choose to end their life |
| Cover Rate | 35.5% |

**C. EXAMPLE OF SETS INCLUDING 5 KEY POINTS**

Table 16 exemplifies a set which includes five key points. This table also shows how many arguments each key point covers.

**D. EXAMPLE OF SETS INCLUDING SIX KEY POINTS**

Table 14 exemplifies a set which includes six key points. This table also shows how many arguments each key point covers.
TABLE 16. Examples of the sets which include five key points for the shown stance in the debate topics. Cover Rate refers to how many arguments key points cover.

| Debate Topic: The vow of celibacy should be abandoned | Stance: Pro | The Number of Arguments: 122 |
|-----------------------------------------------------|------------|-----------------------------|
| Key Point 1: Celibacy leads to wrong sexual behaviour | Cover Rate | 15.6%                       |
| Key Point 2: Celibacy is unhealthy/unnatural          | Cover Rate | 16.4%                       |
| Key Point 3: Celibacy is too pressuring/difficult     | Cover Rate | 8.2%                        |
| Key Point 4: Celibacy is outdated                    | Cover Rate | 37.7%                       |
| Key Point 5: Celibacy reduces the pool of people wanting to become priests | Cover Rate | 5.7%                        |

TABLE 17. Examples of the sets which include six key points for the shown stance in the debate topics. Cover Rate refers to how many arguments key points cover.

| Debate Topic: Assisted suicide should be a criminal offence | Stance: Pro | The Number of Arguments: 125 |
|-----------------------------------------------------------|------------|-----------------------------|
| Key Point 1: Assisted suicide allows people to solicit someone to die to their own benefit | Cover Rate | 9.6%                        |
| Key Point 2: People who are ill do not have clear judgement and are in need of help | Cover Rate | 6.4%                        |
| Key Point 3: People do not have the right to self-harm/harm others | Cover Rate | 5.6%                        |
| Key Point 4: Assisted suicide is akin to killing someone | Cover Rate | 53.6%                       |
| Key Point 5: Assisted suicide violates the doctor’s role | Cover Rate | 4.0%                        |
| Key Point 6: Assisted suicide should not be allowed because many times people can still get better | Cover Rate | 3.2%                        |

REFERENCES

[1] E. Cabrio and S. Villata, “Five years of argument mining: A data-driven analysis,” in Proc. 27th Int. Joint Conf. Artif. Intell., vol. 18, Jul. 2018, pp. 5427–5433.

[2] J. Lawrence and C. Reed, “Argument mining: A survey,” Comput. Linguistics, vol. 45, no. 4, pp. 765–818, Dec. 2020, doi: 10.1162/coli_a_00364.

[8] R. Bar-Haim, I. Bhattacharya, F. Dinuzzo, A. Saha, and N. Slonim, “Stance classification of context-dependent claims,” in Proc. 35th Conf. Eur. Chapter Assoc. Comput. Linguistics, vol. 1, 2017, pp. 251–261.

[9] K. Popat, S. Mukherjee, A. Yates, and G. Weikum, “STANCY: Stance classification based on consistency cues,” in Proc. Conf. Empirical Methods Natural Lang. Process., 9th Int. Joint Conf. Natural Lang. Process., Hong Kong, Nov. 2019, pp. 6414–6419, doi: 10.18653/v1/D19-1675.

[10] N. Reimers, B. Schiller, T. Beck, J. Daxenberger, C. Stab, and I. Gurevych, “Classification and clustering of arguments with contextualized word embeddings,” in Proc. 57th Annu. Meet. Assoc. Comput. Linguistics, 2019, pp. 567–578.

[11] E. M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and N. Das Sarma, “SQUAD: 100,000 sentences and questions for reading comprehension,” in Proc. Conf. Empirical Methods Natural Lang. Process., 9th Int. Joint Conf. Natural Lang. Process., Hong Kong, Nov. 2018, pp. 386–396, doi: 10.18653/v1/N18-1036.

[12] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol., Minneapolis, MN, USA, 1 Jun. 2019, pp. 4171–4186, doi: 10.18653/v1/N19-1423.

[13] M. Persing and V. Ng, “Why Can’t you convince me? Modeling weaknesses in unpersuasive arguments,” in Proc. 26th Int. Joint Conf. Artif. Intell., 2015, pp. 957–966.

[14] R. Bar-Haim, L. Eden, R. Friedman, Y. Kantor, D. Lahav, and N. Slonim, “From arguments to key points: Towards automatic argument summarization,” in Proc. 58th Annu. Meet. Assoc. Comput. Linguistics, 2020, pp. 4029–4039.

[15] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence embeddings using siamese BERT-networks,” in Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP), 2019, pp. 3992–3992.

[16] M. Lippi and P. Torroni, “Context-independent claim detection for argument mining,” in Proc. 24th Int. Joint Conf. Artif. Intell., Jul. 2015, pp. 185–191.

[17] R. Bar-Haim, Y. Kantor, L. Eden, R. Friedman, D. Lahav, and N. Slonim, “Quantitative argument summarization and beyond: Cross-domain key point analysis,” in Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP), Nov. 2020, pp. 39–49, doi: 10.18653/v1/2020.emnlp-main.3.

[18] S. Eger, and I. Gurevych, “ArgumentText: Searching for arguments in heterogeneous sources,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Demonstrations, New Orleans, LA, USA, Jun. 2018, pp. 1–8, doi: 10.18653/v1/N18-5005.

[19] L. Ein-Dor, E. Shnarch, L. Dankin, A. Halfon, B. Sznajder, A. Gera, C. Alfaris, M. Gleize, L. Chosen, Y. Hou, I. Bila, R. Aharonov, and N. Slonim, “Corpus wide argument mining—A working solution,” in Proc. 34th AAAI Conf. Artif. Intell., 2020, pp. 7683–7691.

[20] A. Misra, B. Ecker, and M. Walker, “Measuring the similarity of sentential arguments in dialogue,” in Proc. 17th Annu. Meet. Special Interest Group Discourse Dialogue, Los Angeles, CA, USA, Sep. 2016, pp. 276–287, doi: 10.18653/v1/W16-1636.

[21] M. Lippi and P. Torroni, “Context-independent claim detection for argument mining,” in Proc. 24th Int. Joint Conf. Artif. Intell., Jul. 2015, pp. 185–191.
