Team Enigma at ArgMining-EMNLP 2021: Leveraging Pre-trained Language Models for Key Point Matching

Manav Nitin Kapadnis*, **Sohan Patnaik***,
Siba Smarak Panigrahi*, **Varun Madhavan***, **Abhilash Nandy**
Indian Institute of Technology Kharagpur
{iammanavk, sohanpatnaik106, sibasmarak.p, varun.m.iitkgp, nandyabhilash}@gmail.com

Abstract

We present the system description for our submission towards the Key Point Analysis Shared Task at ArgMining 2021. Track 1 of the shared task requires participants to develop methods to predict the match score between each pair of arguments and keypoints, provided they belong to the same topic under the same stance. We leveraged existing state of the art pre-trained language models along with incorporating additional data and features extracted from the inputs (topics, key points, and arguments) to improve performance. We were able to achieve *mAP* strict and *mAP* relaxed score of 0.872 and 0.966 respectively in the evaluation phase, securing 5th place[1] on the leaderboard. In the post evaluation phase, we achieved a *mAP* strict and *mAP* relaxed score of 0.921 and 0.982 respectively. All the codes to generate reproducible results on our models are available on Github[2].

1 Introduction

The Quantitative Summarization - Key Point Analysis (KPA) Shared Task requires participants to identify the keypoints in a given corpus. Formally, given an input corpus of relatively short, opinionated texts focused on a particular topic, KPA aims to identify the most prominent keypoints in the corpus. Hence the goal is to condense free-form text into a set of concise bullet points using a well-defined quantitative framework. In track 1, given a debatable topic, a set of keypoints per stance, and a set of crowd arguments supporting or contesting the topic, participants must report for each argument the corresponding match score for each keypoint under the same stance towards the topic. In track 2, we are required to build a language model that would generate keypoints given a set of arguments and a topic and finally find the match score of that particular keypoint with the argument. We mainly focused on the first track.

We frame the task of identifying the most prominent keypoints as a sentence similarity task, obtaining the most similar keypoints corresponding to a given argument.

2 Related Work

Sentence similarity is gaining much attention in the research community due to its versatility in various natural language applications such as text summarization (Abujar et al., 2019), question answering (Ashok et al., 2020), sentiment analysis (Khamphakdee and Seresangtakul, 2021) and plagarisim detection (Lo and Simard, 2019). Two major approaches to quantitatively measure similarity have been proposed -

- **Lexical similarity**, as the name suggests, is a measure of the extent or degree of lexicon overlap between two given sentences, ignoring the semantics of the lexicons.
- **Semantic similarity** takes into account the meaning or semantics of the sentences. Deep Learning based approaches are typically leveraged to create dense representations of sentences, which are then compared using statistical methods like cosine similarity.

Since the *ArgKP-2021* dataset (Friedman et al., 2021) contains crowd arguments for or against a particular stance, naturally, we expect some paraphrasing in the arguments put forth by different people. This indicates that semantic similarity would be an appropriate measure of similarity. However, we observe the problem of semantic drift (Jansen, 2018) in keypoint - argument pairs. Hence, we add additional lexical overlap and syntactic parse based features to improve performance (details on the features can be found in Section 4).

*Equal contribution.

[1] All results and leaderboard standings are reported using the default evaluation method (explained in section 5)

[2] https://github.com/manavkapadnis/Enigma_ArgMining

Proceedings of The 8th Workshop on Argument Mining, pages 200–205
Punta Cana, Dominican Republic, November 10–11, 2021. ©2021 Association for Computational Linguistics

3 Dataset Description

The *ArgKP-2021* dataset (Friedman et al., 2021) which was the main dataset used for the shared task consists of approximately 27,520 argument/keypoint pairs for 31 controversial topics. Each of the pairs is labeled as matching or non-matching, along with a stance towards the topic. The train data comprises of 5583 arguments and 207 keypoints, the validation data comprises of 932 arguments and 36 keypoints and the test data comprises of 723 arguments and 33 keypoints.
Additionally, since external datasets were permitted, we experimented with two more datasets i.e., the IBM Rank 30k dataset (Gretz et al., 2019) and the Semantic Textual Similarity or STS dataset (Cer et al., 2017) (described in section 4.5) to train our model before fine-tuning on the *ArgKP-2021* dataset. The *STS* dataset comprises of 8020 pairs of sentences, whereas the IBM Rank 30k dataset comprises of 30497 pairs of arguments and keypoints.

4 Implementation Details

In this section, we elaborate on our experiments and methodology to find the best-performing models. The section is organized to describe the addition of dependency parsing features in Section 4.2, parts of speech features in Section 4.3, Tf-idf features in Section 4.4, and the use of external datasets in Section 4.5.

4.1 Baseline Transformer Model Architecture

In recent work, Transformer (Vaswani et al., 2017) based pre-trained language models like BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), BART (Lewis et al., 2019), and DeBERTa (He et al., 2021), have proven to be very powerful in learning robust context-based representations of lexicons and applying these to achieve state of the art performance on a variety of downstream tasks.

We leverage these models for learning contextual representations of a keypoint - argument pair. The keypoints and arguments are individually concatenated, along with the topic (in the same order) for additional context information. We then obtain the contextual representation of this triplet and concatenate to it an encoded feature vector of additional features (one of Dependency Parse based features, Parts-of-Speech based features, and Tf-idf vectors). This concatenated vector was then passed through dense layers and a sigmoid activation to get a final similarity score in the desired range of $[0, 1]$, as shown in Figure 1.

4.2 Dependency Parsing Features

To capture the syntactic structure of the sentences, we added the dependency parse tree of the sentence as an additional feature.

To obtain the same, we used the open-source tool *spacy* [3]. The dependency features are then label encoded according to descending order of occurrences. Consider three unique dependency features in all the concatenated sentences of the original dataset, namely, 'aux', 'amod', and 'nsubj'. Let 'aux', 'nsubj', and 'amod' be the descending order of count in the dataset, then 'aux' is encoded as one, 'nsubj' as two and 'amod' is encoded as three. All the names of unique features can be found in the supplementary material.

These encoded dependency features are then concatenated to the output of the transformer model and passed to subsequent layers as shown in Figure 1.

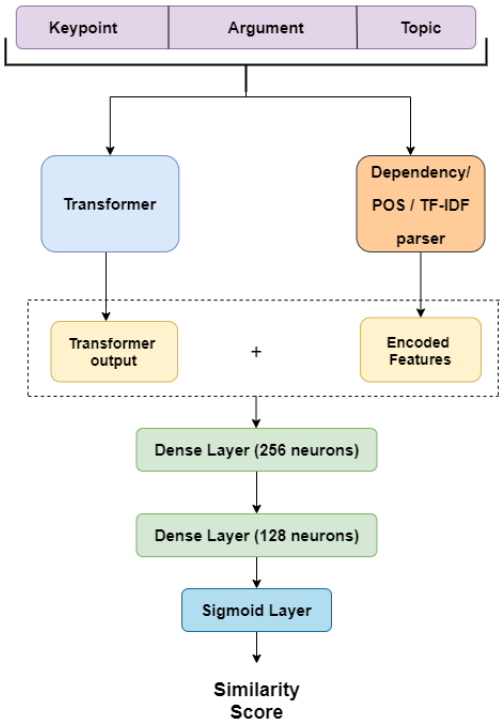

Figure 1: Model Architecture ("+" implies concatenation)

4.3 Parts of Speech Features

With a similar motive as before, i.e., to better capture the syntactic structure of the sentences, we

[3] https://spacy.io/

experimented with Part-Of-Speech (POS) Features as well.

As before, we used the open-source tool *Spacy* to obtain POS labels for each lexicon, which were then label encoded according to descending order of occurrences. The encoded feature vector is then concatenated to the output of the transformer model and fed to the subsequent layers.

4.4 Tf-idf features

In addition to semantic overlap, we wished to see if adding lexical overlap-based features would improve the ability of the model to identify similar sentences. To this end, we obtained the Tf-idf vector of the (keypoint, argument, topic) triplet (with padding). As before, the encoded feature vector is then concatenated to the output of the transformer model and fed further to the subsequent layers.

4.5 External Datasets

We further tried to experiment with sentence similarity pre-training task on two additional datasets. The two datasets used were the STS benchmark dataset and the IBM Debater® - IBM Rank 30k dataset.

For the STS dataset, we normalized the target similarity score to bring the scores between 0 and 1. No additional preprocessing was done to the text. The two input sentences were concatenated into a single sentence and then directly fed to the model. We trained our model on STS dataset for 6 epochs and on the main dataset for 3 epochs.

For the IBM Rank 30k dataset, we used the MACE (Hovy et al., 2013) Probability score as the target column, which signifies the argument quality score for the corresponding topic. This is analogous to our approach for main task, wherein we output a similarity score for each argument-keypoint pair. No preprocessing was done to the text, the argument and topic were concatenated into a single sentence and then fed to the model. We trained our model on the IBM Rank 30k dataset for 3 epochs and on the main dataset for 3 epochs.

Due to resource constraints, we were not able to perform pre-training on both the additional datasets one after another.

5 Results and Discussions

After we had concluded our experiments, a new evaluation method was proposed by organizers, which removes the positive bias towards a system that predict less true positives in high confidence. In the default evaluation metric a perfect recall is attained only when all positive ground truth labels are predicted, whereas the new method allows a perfect recall score when the top 50% of the predictions (ranked by confidence) are positive. However, since we had completed all our experiments at this point, it was not feasible to rerun all our experiments in the given time frame. Hence we have reported all our results according to the default evaluation method.

Among all the transformer models without the use of external datasets, we found BART-large to perform best, along with DeBERTa-large with Tf-idf as additional features, achieving the best *mAP* strict and *mAP* relaxed score of 0.909, 0.982 and 0.911, 0.987 respectively. All the reported results are averaged over three seeds.

Table 1 describes our experiments with different Transformer-based contextual language models without using any additional features. Recent improvements to the state-of-the-art in contextual language models in BART and DeBERTa perform significantly better than BERT. Further, BART is pre-trained using various self-supervised objectives such as token masking, sentence permutation, document rotation, token deletion and text infilling, unlike other models that mostly use either masked language modelling or next sentence prediction. In our opinion, the tasks of sentence permutation and document rotation help the model get a better understanding of context at the sentence level, and thus, are helpful when considering the keypoint matching task. We also observe that the *large* version of the models, trained on more data with more parameters, perform significantly better than the *base* versions, as expected.

Model	mAP Strict	mAP Relaxed
BERT-base	0.804 ± 0.037	0.910 ± 0.050
RoBERTa-base	0.826 ± 0.051	0.930 ± 0.032
BART-base	0.824 ± 0.030	0.908 ± 0.020
DeBERTa-base	0.894 ± 0.020	0.973 ± 0.015
BERT-large	0.821 ± 0.025	0.924 ± 0.006
RoBERTa-large	0.892 ± 0.003	0.970 ± 0.015
BART-large	$\mathbf{0.909 \pm 0.011}$	$\mathbf{0.982 \pm 0.003}$
DeBERTa-large	0.889 ± 0.030	0.979 ± 0.010

Table 1: Results of Transformer models

Table 2 shows the best performing results obtained by concatenating one of the following - Dependency Parse features, POS features, and Tf-idf features. We note that out of the three feature

vectors methods, Tf-idf features performs the best. Tf-idf gives a relation/measure of lexical overlap between the argument and keypoint, whereas the other features (POS and Dependency Parse) just expand on the sentence structures of the argument and the keypoint, without expressing the relation between the same. Thus it is observed that Tf-idf performs better than the other two feature vectors. In table 2, we report the best-performing transformer-based models for each feature vector. Detailed results (each transformer model with each feature) can be found in the Appendix which is present in the supplementary material. We could not perform combination of all the syntactic features due to limited GPU memory availability.

Feature	Best Model	mAP Strict	mAP Relaxed
Dep[4]	BART-large	0.868 ± 0.023	0.977 ± 0.015
POS[5]	BART-large	0.906 ± 0.011	0.987 ± 0.005
Tf-idf	DeBERTa-large	**0.911 ± 0.005**	**0.987 ± 0.008**

Table 2: Results with Additional Features

Table 3 shows the outcome of training on additional datasets such as the STS and the IBM Rank 30k dataset without using any feature vectors. We find that the best performing scores using both these datasets are almost equal and are achieved by the same BART-large model architecture. Thus training on additional datasets led to a substantial increase in both *mAP* strict and *mAP* relaxed scores. The best results of pre-training on the additional datasets were almost similar, which might be because the ground truth scores in both the datasets effectively reflect the semantic overlap between two sentences (i.e., if two sentences of a data sample are semantically similar, they would have a higher score, and vice versa), thus making the datasets similar to one another.

We also tried adding feature vectors plus training on additional datasets[6], but there was no significant change in the performance than the existing results. Transformers themselves are able to learn syntactic and semantic features on their own during the training process (Clark et al., 2019). Adding these features only increases redundancy, as a result of which the performance of the model isn't affected much. This observation could also be seen in the difference in the results of table 1 and 2.

[4]Encoded dependency features (section 4.2)

[5]Encoded parts of speech features (section 4.3)

[6]The results of these experiments can be found in Appendix available in the supplementary material.

Complete results of these experiments can be found in the Appendix available in the supplementary material.

Model	Additional Dataset	mAP Strict	mAP Relaxed
BERT-large	STS	0.818 ± 0.045	0.933 ± 0.016
RoBERTa-large	STS	0.905 ± 0.007	**0.986 ± 0.004**
BART-large	STS	**0.920 ± 0.005**	0.967 ± 0.036
DeBERTa-large	STS	0.912 ± 0.004	0.983 ± 0.003
BERT-large	IBM Rank 30k	0.793 ± 0.029	0.914 ± 0.019
RoBERTa-large	IBM Rank 30k	0.872 ± 0.006	0.974 ± 0.003
BART-large	IBM Rank 30k	**0.921 ± 0.018**	**0.982 ± 0.002**
DeBERTa-large	IBM Rank 30k	0.894 ± 0.017	0.982 ± 0.008

Table 3: Results with pretraining on additional datasets

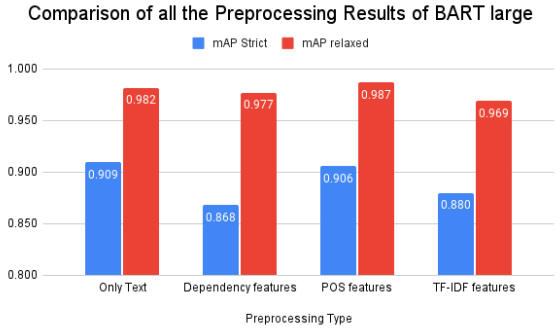

Figure 2: All preprocessing methods with BART large

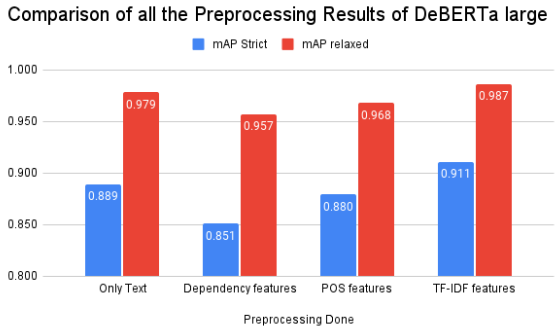

Figure 3: All preprocessing methods with DeBERTa large

In Figures 2 and 3, we plot the results of the best-performing transformer-based models using different feature vectors.

6 Ablation Study

We designed different settings to compare and validate our approach and its performance. This section consists of results on excluding of topics from input in Section 6.1, incorporating average of hidden states before feeding to dense layers in Section 6.2, and boosting in Section 6.3. Since we

obtain best results with BART-large and DeBERTa-large with Tf-idf features, thus the following ablation study is done with these class of models.

6.1 Exclusion of topic from input

We incorporate the combination of keypoints and arguments as input to the pre-trained language models to analyze the importance of the topic towards generating the matching score. Comparing Table 1 and Table 4, incorporating topic provides more context in the input, thus improving both *mAP* strict score and *mAP* relaxed score.

Model	mAP Strict	mAP Relaxed
BART-base	0.803 ± 0.028	0.898 ± 0.015
DeBERTa-base	0.823 ± 0.030	0.922 ± 0.012
BART-large	$\mathbf{0.880 \pm 0.006}$	$\mathbf{0.946 \pm 0.010}$
DeBERTa-large	0.874 ± 0.025	0.946 ± 0.027

Table 4: Results with input as keypoint plus argument

6.2 Average of hidden states

We average the last two and the last three hidden states of the pre-trained language model. The average hidden states were then fed into the dense layers to obtain the match score. It can be observed that for both BART-large and DeBERTa-large, the performance decreases as we incorporate more hidden states for the output. The intuition behind this observation can be attributed to the fact that task-specific information encoded in hidden states is less as compared to the last layer, resulting in decreased performance. The results are shown in Table 5.

Model	No. of Hidden States	mAP Strict	mAP Relaxed
BART-large	2	0.868 ± 0.016	0.941 ± 0.004
DeBERTa-large	2	$\mathbf{0.871 \pm 0.039}$	$\mathbf{0.949 \pm 0.015}$
BART-large	3	0.837 ± 0.020	0.933 ± 0.012
DeBERTa-large	3	0.850 ± 0.014	0.934 ± 0.022

Table 5: Results with average of hidden states

6.3 Boosting

We implemented the AdaBoost algorithm by considering our baseline transformer architecture as the base model for this sequential paradigm. BART-large and DeBERTa-large were the transformers used for this study. The first base model was trained with the whole training set, whereas the other four models were trained by sampling data points from a probability distribution. Initially, all the data points were assigned an equal probability. However, the distribution was updated in a way such that the erroneous data points for the previous base models were given a higher probability to be sampled.

The top $10,000$ most probable data points were sampled for each base model except for the first one. It can be observed from Table 1 and Table 6 that for DeBERTa large model, the *mAP* Strict has indeed been boosted from 0.889 to 0.904. The results are mentioned in Table 6.

Model	mAP Strict	mAP Relaxed
BART-large	0.832 ± 0.020	0.960 ± 0.010
DeBERTa-large	$\mathbf{0.904 \pm 0.021}$	$\mathbf{0.973 \pm 0.017}$

Table 6: Boosting Results on Transformer model

7 Conclusion

In this work, we used Pre-trained Language Models (PLMs) to predict the match score for each argument and keypoint pair under the same stance towards the topic. We observed the state-of-the-art PLMs such as BART and DeBERTa perform the best compared to other models. We further improve the performance with additional datasets (IBM Rank 30k and STS) to perform additional pre-training (with sentence similarity) before fine-tuning on ArgKP-2021 dataset. We experimented with POS, Dependency and Tf-idf features to evaluate the addition of extra syntactic features. We support the selection of our final models with various ablation studies. It would be a good future direction to generate appropriate explanations from concatenated input and propose methods to use explanations in the training process.

8 Acknowledgements

We would like to thank the organizers Roni Friedman-Melamed, Lena Dankin, Yufang Hou, and Noam Slonim for holding this shared task. It was a great learning experience for us. We would also like to thank our fellow participants at ArgMining 2021; we look forward to learning more about their approaches and interacting with them at EMNLP. Finally, we would like to extend a big thanks to makers and maintainers of the exemplary HuggingFace (Wolf et al., 2020) repository, without which most of our research would have been impossible.

References

Sheikh Abujar, Mahmudul Hasan, and Syed Akhter Hossain. 2019. Sentence similarity estimation for text summarization using deep learning. In *Proceedings of the 2nd International Conference on Data Engineering and Communication Technology*, pages 155–164, Singapore. Springer Singapore.

Aishwarya Ashok, Ganapathy Natarajan, Ramez Elmasri, and Laurel Smith-Stvan. 2020. SimsterQ: A similarity based clustering approach to opinion question answering. In *Proceedings of The 3rd Workshop on e-Commerce and NLP*, pages 69–76, Seattle, WA, USA. Association for Computational Linguistics.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does bert look at? an analysis of bert's attention.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. 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.*

Shai Gretz, Roni Friedman, Edo Cohen-Karlik, Assaf Toledo, Dan Lahav, Ranit Aharonov, and Noam Slonim. 2019. A large-scale dataset for argument quality ranking: Construction and analysis.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130, Atlanta, Georgia. Association for Computational Linguistics.

Peter Jansen. 2018. Multi-hop inference for sentence-level TextGraphs: How challenging is meaningfully combining information for science question answering? In *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12)*, pages 12–17, New Orleans, Louisiana, USA. Association for Computational Linguistics.

Nattawat Khamphakdee and Pusadee Seresangtakul. 2021. A framework for constructing thai sentiment corpus using the cosine similarity technique. In *2021 13th International Conference on Knowledge and Smart Technology (KST)*, pages 202–207.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

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.

Chi-kiu Lo and Michel Simard. 2019. Fully unsupervised crosslingual semantic textual similarity metric based on BERT for identifying parallel data. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 206–215, Hong Kong, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.