Key Point Matching with Transformers

Emanuele Cosenza
University of Pisa, Italy
e.cosenza3@studenti.unipi.it

Abstract

Key point analysis (KPA) has been proposed as a way to summarize arguments with short-sized pieces of text termed key points. This work aims at describing a solution for the Track 1 of the KPA 2021 shared task, analyzing different methodologies for the specific problem of key point matching, which consists in finding a reasonable mapping from arguments to key points. The analysis will focus on transformer based architectures, experimentally investigating the effectiveness of variants specifically tailored to the task.

1 Introduction

In the context of natural language processing, text summarization is the act of creating a concise summary of a given text. This is particularly useful when the text at hand represents arguments, i.e. opinions, about a certain topic. An example is that of political debates, in which politicians express opinions concerning the subject matter. Another scenario is that of e-commerce sites, where users can write reviews about different items. In both cases, a third party can greatly benefit from a summary of the arguments in order to make effective decisions.

In recent years, there have been several attempts to automatize the summarization of arguments in a meaningful way. The first works focused on clustering, by which arguments are assigned to clusters based on different notions of relatedness (e.g. semantic similarity) between pairs of arguments (Reimers et al., 2019; Ajjour et al., 2019; Misra et al., 2016). Unfortunately, with this methodology the actual summary of the arguments had to be compiled manually by analyzing the contents of the clusters. More recently, Bar-Haim et al. (2020a) proposed to transform the summarization problem into what has been named key point analysis (KPA). Given a certain topic of discussion, arguments are summarized by matching them to a short list of high-level summaries, called *key points*. This allows a quantitative approach to summarization, in which the saliency of a key point (i.e. the degree of matched arguments) gives a hint of the underlying nature of the arguments. Even though Bar-Haim et al. (2020a) have shown that a domain expert can create meaningful key points for a topic even without looking at the arguments, the final goal of this field would be to generate the key points and match them with the respective arguments without human intervention. With this goal in mind, the KPA 2021 shared task [1] (Friedman et al., 2021) poses two problems to its participants: a key point matching subtask (Track 1) and a key point generation subtask (Track 2). This paper describes a solution to the key point matching subtask, in which key points are given and the goal is to match them with the appropriate arguments. The problem is framed as a classification task and the paper will focus on analyzing different transformer based models with different possible variants.

All experiments are executed on a Nvidia Tesla P100 with limited access. The computational limitations have influenced many design choices and results of the paper.

2 KPA 2021 shared task: data and evaluation metrics

As training data, the proponents shared the *ArgKP* dataset (Bar-Haim et al., 2020a) divided into training and development sets, containing respectively 24 and 4 debate topics of various nature and arguments mined from the internet. The dataset contains 24,093 `<argument, key point>` pairs. Each pair is associated with the concerned topic, presented as a short-sized text, and the stance towards the topic (+1 for the pro side, -1 for the con side). A binary label (1 for a match, 0 otherwise) is

[1] https://github.com/IBM/KPA_2021_shared_task

Proceedings of The 8th Workshop on Argument Mining, pages 190–199
Punta Cana, Dominican Republic, November 10–11, 2021. ©2021 Association for Computational Linguistics

assigned only to <argument, key point> pairs under the same stance towards the topic, noting that arguments and key points having opposite stance automatically constitute a mismatch. At the end of the competition, the task organizers have also released an external test set containing 3 topics, totalling 3426 pairs, to be used for the final evaluation and ranking of participants.
Not every feasible pair appears in the labeled data. A pair appears in the dataset as a match or a mismatch if, respectively, more than 60% or less than 15% of the original annotators labeled it as matching. All the other pairs are not included in the dataset. Intuitively, this ensures that the pairs are not labeled ambiguously whenever there was no net agreement between the annotators. Even if the dataset is constructed like so, in the evaluation phase of the task the final model will be queried on all possible pairs including the unlabeled ones, testing the system's ability to deal with uncertainty. In fact, the model will have to output, for each argument, a match score in [0, 1] for each of the key points under the same topic and in the same stance towards the topic.

The task is evaluated by the average rank of the *strict* and the *relaxed* mean Average Precisions (mAP), where the ranks are intended as the final positions of the two values in the rankings of the competition. The mAP, which is the mean of the precision values of the model at different recalls, is chosen as a metric since it gives a summary of the recall-precision curve without having to choose a specific classification threshold. Given the match scores output by a model on a certain dataset, the strict and relaxed mAP values are computed as follows. First, each argument is paired with the highest scoring key point. Then, half of the lowest matching pairs are discarded. This is done to avoid evaluating the model on arguments with no matching key points. At this point, precision values and therefore mAP values can be computed on the remaining pairs based on the labeled data. Note, however, that a pair can have a high match score but no labels in the dataset because of its ambiguity. In this case, the pair can arbitrarily be considered a match or a mismatch. This results in the computation of two different mAP scores: a strict mAP, for which absent pairs are considered a match, and a relaxed mAP, for which the pairs are considered a mismatch. The final values are computed by taking the macro-average of strict and relaxed mAP values for each topic/stance combination.

3 Methodological survey

Assigning a match score to an <argument, key point> pair can be considered as a text classification task where the problem is to determine the probability that the pair is a match. In recent years, transfer learning through fine-tuning of pretrained transformers such as BERT (Devlin et al., 2018) have dramatically improved results on a variety of NLP tasks, including classification. Furthermore, transformer based architectures are particularly suited for the task of sentence pair classification. In fact, many of these models are pretrained on tasks such as next sentence prediction (NSP) and sentence order prediction (SOP), which require the model to take in pairs of sentences or segments and to classify them accordingly. This implies that the *ArgKP* dataset can be easily adapted to these models. Moreover, since the models have already learned how to process and internally represent sentences, one would expect to see good results also in the context of key point matching. For these reasons, this work focuses on transformers. This is not a limiting decision, since there are many transformers architectures to test and many possible variants for each model. In literature, previous works related to KPA (Bar-Haim et al., 2020a,b) already report results for 4 transformer based models, namely BERT (Devlin et al., 2018), XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019) and ALBERT (Lan et al., 2019), considering only their largest versions. For each model, only classification metrics such as accuracy, precision, recall and F1 scores are reported. This work aims at expanding the analysis by studying the behavior of the models under the metrics of the competition and by experimentally evaluating new variants. In particular, to allow a comparison with the state of the art, the same 4 transformers are analyzed also here, extending the investigation to smaller versions of the models and to other variants that will be described later on. Smaller models are not expected to be superior to the larger versions. Nonetheless, they are included in the study to give a reference point regarding their performance on the task of key point matching.
Key point matching can also be considered as an instantiation of a natural language inference (NLI)

task (Dagan et al., 2013), where the problem is to determine whether the key point logically follows from the argument (or viceversa). However, fine-tuning models pretrained on NLI datasets already showed no improvement in literature (Bar-Haim et al., 2020a). Because of this and of the additional computational burden caused by pretraining on large NLI datasets, this variant is avoided.

Besides basic transformers, this work includes SBERT (Reimers and Gurevych, 2019), a sentence embedding transformer model. SBERT produces semantically meaningful sentence embeddings, which means that semantically similar sentences are close in the vector space. SBERT has been used alone, without fine-tuning, computing the cosine similarity between arguments and key points embeddings. Furthermore, it has also been used in one of the variants, feeding its outputs to the linear top-layer of the transformer based models to see if adding information about semantically meaningful sentence embeddings yields an increase in the performance.

BERT, XLNet, RoBERTa and ALBERT have been downloaded through the huggingface library (Wolf et al., 2020), while SBERT has been downloaded through the Sentence-Transformers Python framework, choosing the `paraphrase-mpnet-base-v2` model since it has the best performance on a variety of different tasks. BERT is uncased. XLNet and RoBERTa are cased since no uncased versions have been released for them. For each transformer other than SBERT and ALBERT, both `base` and `large` versions have been tested. For what concerns ALBERT, `albert-base-v2` has been used for the smaller version, while `albert-xxlarge-v1` has been chosen for the larger version since it is generally better than `albert-xxlarge-v2` on downstream tasks.

A thorough analysis should have included also RNNs and CNNs. In fact, arguments and key points are generally short-sized and simpler models may actually work better in this context. However, given the non negligible computational limitations of this work, discarding these and other valuable models allowed for the exploration of a greater number of transformer based architectures and variants.

4 Data preprocessing and analysis

Since the training data shared by the organizers is split into a training (TR) and a validation set (VL), an internal test set (TS) split had to be extracted from the TR in order to assess the performance of the final model. In order not to allow different `<argument, key point>` pairs under the same topic to be spread between different splits, the TS has been extracted from the TR by doing a topic based division. The final internal TS contains 3 topics with a total of 3164 pairs, while the final TR and VL contain respectively 21 and 4 topics with a total of 17471 and 3458 pairs.

Arguments are mined from the internet and even if they have been originally filtered, their quality can vary a lot throughout the dataset. They sometimes contain misspellings and an informal use of symbols (e.g. slash between words). Even if there is evidence of the fact that BERT based models can suffer from natural adversarial instances such as typos (Sun et al., 2020), the quality of the dataset has not been improved for the sake of simplicity.

As mentioned earlier in the paper, for each `<argument, key point>` pair the dataset also contains the concerned topic and the stance towards it. Adding the topic to the input of the model is useful especially when neither the argument nor the key point contains enough information about the context of discourse. In previous works, the topic has not been included in the inputs and therefore its utility will be tested in one of the variants presented later. When the topic is considered, it is simply prepended to the concerned argument and the input to the model therefore becomes `<topic+argument, kp>`. Notice, however, that topics are always written in the affirmative form as in *"Kids should be wearing uniforms at school"*. When the stance towards the topic is negative, in order not to fool the model by giving it contradictory information, the topic should be expressed in its negative form. One way to do this would be to encode the stance as a binary input to the linear output layers. Here, instead, affirmative topic are simply transformed into their negative form in order to exploit the encoding capabilities of the transformers. Transforming an affirmation to a negation is not a trivial task, but fortunately the topics contained in the dataset and in the external TS are all in the form SUBJECT + *should* or *is* + REST. This allows to just add *not* after *should* or *is* in order to complete the transfor-

mation, noting that, in order to make the system generalizable, one would have to develop a fully working affirmative-to-negative module. The alternative to this would have been to encode the stance as a binary input to the models.
Arguments appear in the dataset in all lower case, all caps (a very small portion) and a mixture of both. Arguably, in the context of key point matching, a cased analyis of the text would not be much useful. For this reason, `uncased` models have been used when possible, transforming the text to lowercase before processing it. All text is tokenized and transformed to lowercase by relying on the different tokenizers provided by each transformer. Finally, each `<arg, kp>` pair is transformed by the tokenizer as follows: `[cls] arg [sep] kp [sep]`.

5 Models and variants

In the model selection phase, BERT, ALBERT, ROBERTA, XLNet and SBERT are all tested with the following modalities. For each model except SBERT, the representations computed by the transformers are fed to a single linear layer with two output logits. A softmax layer is then used at evaluation time to compute the final match probability. Each model except SBERT can have multiple possible configurations corresponding to different variants of the basic models. This work focuses on three main variants. First, one can decide whether to inform or not the model about the topic with the modalities shown in the previous section. Second, the final contextualized representations output by the transformers can be computed in a number of different ways as in Devlin et al. (2018). This was allowed since the best representation usually depends on the particular task at issue. One can decide to just take the `[cls]` token representation or to get a final vector by applying a pooling operation to the hidden representations of the transformer. Three pooling options are available: average of the last layer (`avg_last`), average of the second last layer (`avg_2ndlast`), average of the sum of the last four layers (`avg_sumlast4`). Finally, one can choose to concatenate the 768-dimensional SBERT embeddings of the two input segments to the representation output by the transformer. This was allowed to see if adding information about semantically meaningful sentence embeddings would help the models in the classification task.

6 Training procedure and model selection

6.1 Metrics

Before training, an evaluation metric had to be defined in order to make the model selection possible. The metric should follow the one used in the competition so to favor models that can potentially place high in the final rankings. As explained earlier, the metrics involved in the competition are the strict and relaxed mAP scores. More specifically, the final rank of a system is determined by the average of the ranks of the strict and relaxed mAP values. For this reason, the average of the two scores, indicated from now on with Avg mAP, has been used as the metric. Notice that this is not the same thing as the average of the ranks. Still, this is a reasonable way give importance to both scores. Furthermore, a model with a high Avg mAP is also likely to have a high average rank. When computing the Avg mAP, the strict and relaxed mAP scores have been determined with the already outlined procedure used by the organizers.

Even if the task does not require the models to compute the actual classification labels, for the sake of completeness, classification metrics such as accuracy, precision, recall and F1 scores have been computed both on the TR and the VL. Computing classification labels from match scores can be done in multiple ways as shown in Bar-Haim et al. (2020a). It can be done locally, considering just the single score of a pair, or globally, taking into account all the match scores between an argument and the key points. Since this problem was not the focus of the paper, only two simple local methods have been evaluated. In the first one, a match score is trivially transformed using a threshold of 0.5. In the second one, the threshold is learned and is chosen to maximize the F1 score on the TR. Since the learned threshold often generalizes worse than the fixed one, it is not considered in the discussion and related results are shown in Appendix A.

6.2 Training

All models are fine-tuned end to end without freezing the transformers, using Adam as the optimizer and cross entropy loss as the loss function. The hyperparameters of the models are fixed before the model selection, following when possible the configurations employed in the papers of the respective transformers. The dropout rate of

the transformers is always the default one (0.1). In the final linear layer, dropout is always set to 0.1. Also, the maximum number of epochs e is always 3 and checkpoints are saved in each epoch whenever the cross entropy loss on the VL reaches a new minimum. The number of epochs was found by manually checking the convergence of the models during the preliminary phase of the study. The batch size b is set to 32. When training smaller models a fixed learning rate is used. For the largest models, which show no convergence with a constant learning rate, a learning rate schedule is used following the original papers. The same warm up and linear decay schedules are used for each model. The duration of the warm up phase was found by trying different values and by checking if the models converged. At the end, the number of training steps of the warm up phase was heuristically computed as $\frac{1}{5}$ of the total training steps $\frac{|TR|}{b}e$, which amounts to around 300 steps.

6.3 Model selection

For computational reasons, the model selection procedure has been divided into two phases. The procedure is suboptimal but it nonetheless gives a hint of what is the appropriate model for the task. For the same reason, instead of a robust cross-validation procedure, hold-out validation is executed in both phases. Given the relatively high number of models to be tested and the long training time of transformers, cross-validation would have been unfeasible with the computational limitations of this work. In the first phase, the chosen models (both small and large) are all trained and validated with their basic configurations, which are the ones that use the `cls` token representation, no topic and no concatenation of SBERT embeddings. Then, the best model on the VL, i.e. the one that has the highest Avg mAP score, is selected for the second phase. In this phase, the selected model is trained again on the same data splits in a grid search with 16 different configurations. At the end, the model with the best configuration on VL is chosen as the final system and is used for the evaluation on the internal and external TS.
Given the limited access to the used machine, all results are obtained with a single random seed, without averaging multiple runs.

6.4 Results

Results on TR and VL of the first phase of the model selection are shown in Table 1. As already mentioned, training is executed only on labeled data, while results are computed also on all possible <arg, kp> pairs. In the table, strict and relaxed mAP values computed on all possible pairs are shown on VL but not on TR. This is done to avoid showing TR results on data which the model has never seen during training. The table also contains the average of strict and relaxed mAP. This is the metric used to select the best model. SBERT, the only unsupervised model in the analysis, is the worst model as expected and serves as a baseline for the other models. However, even if the model is not fine-tuned for the task, the results are not so far from those of the fine-tuned models in terms of Avg mAP. This is remarkable, given that the sentence embeddings and the cosine similarity scores can be computed in a matter of seconds without any training process. RoBERTa, on the other hand, is the best of the models in terms of Avg mAP with ALBERT as a close second. Table 2 shows the classification metrics on TR and VL for the same models. The results are similar to those reported in Bar-Haim et al. (2020a). ALBERT is the best model under all metrics except for the recall, while SBERT is the worst one with a very low precision, showing its weaknesses as an unsupervised model. Even if RoBERTa managed to beat the other models under the specified metrics, ALBERT may be better overall for the task of key point matching when considering non ambiguous data. However, given the results of this first model selection, RoBERTa was selected for the second phase.

Results on both TR and VL for the second phase of the model selection are shown in Table 3. Remember that with the employed hold-out validation procedure results may be non representative of the actual performance of the models. Nonetheless, analyzing the general behavior of the different variants may give an idea about their effectiveness. Variants exploiting the topic show an average increase of more than 0.02 in the Avg mAP, indicating that including the topic in the input actually helped. On the other hand, there are no clear winners among the different pooling methods. On average, `avg_last` was the best, while `cls` was the worst. Similarly, the concatenation of the SBERT embeddings did not make much difference and on average they actually caused a

Model	Type	TR	VL				
		Loss	Loss	mAP (S)	mAP (R)	Avg mAP	Time (TR)
SBERT	paraphrase-mpnet-base-v2	0.743	0.729	0.759	0.904	0.831	-
BERT	base-uncased	0.115	0.409	0.817	0.964	0.891	00:06:10
	large-uncased	0.127	0.389	0.765	0.95	0.857	00:20:41
XLNet	base-cased	0.27	0.302	0.779	0.94	0.86	00:09:41
	large-cased	0.207	0.333	0.767	0.945	0.856	00:27:38
RoBERTa	base	0.153	0.271	0.823	0.954	0.889	00:06:17
	large	**0.057**	0.226	**0.884**	**0.995**	**0.939**	00:21:17
ALBERT	base-v2	0.134	0.298	0.762	0.926	0.844	00:06:33
	xxlarge-v1	0.09	**0.223**	0.879	0.992	0.935	02:03:37

Table 1: Results and training times of transformer based models on TR and VL in the first phase of the model selection. (S) = strict, (R) = relaxed, Loss = cross entropy loss. Avg mAP is the average between strict and relaxed mAP.

	Model	Type	Accuracy	Precision	Recall	F1
TR	SBERT	paraphrase-mpnet-base-v2	0.455	0.261	0.888	0.403
	BERT	bert-base-uncased	0.963	0.911	0.909	0.91
		bert-large-uncased	0.96	0.911	0.893	0.902
	XLNet	xlnet-base-cased	0.894	0.753	0.73	0.741
		xlnet-large-cased	0.918	0.839	0.75	0.792
	RoBERTa	roberta-base	0.945	0.929	0.793	0.856
		roberta-large	**0.98**	**0.981**	0.919	**0.949**
	ALBERT	albert-base-v2	0.956	0.968	0.813	0.884
		albert-xxlarge-v1	0.971	0.926	**0.937**	0.931
VL	SBERT	paraphrase-mpnet-base-v2	0.488	0.284	0.921	0.434
	BERT	bert-base-uncased	0.82	0.554	0.82	0.661
		bert-large-uncased	0.824	0.561	0.813	0.664
	XLNet	xlnet-base-cased	0.891	**0.771**	0.695	0.731
		xlnet-large-cased	0.865	0.676	0.709	0.692
	RoBERTa	roberta-base	0.896	0.764	0.745	0.754
		roberta-large	0.912	0.744	0.894	0.812
	ALBERT	albert-base-v2	0.89	0.727	0.774	0.75
		albert-xxlarge-v1	**0.918**	0.749	**0.925**	**0.828**

Table 2: Classification results on TR and VL with classification threshold set to 0.5 in the first phase of the model selection.

slight degradation of the performance. Finally, the model with the best Avg mAP score was the one exploiting the [cls] token representation, the topic and the SBERT embeddings. This is the final model used for the evaluation on the internal and external TS. Table 4 shows its results under the classification metrics on TR and VL.

7 Test set results and analysis

Having chosen the best model on the VL, the last phase consisted of its assessment on the internal and external TS. Results on both internal and external TS with a reference to previous results on VL are shown in Table 5. Under the chosen metrics, the performance of the model on the internal TS is remarkably higher than on the VL, showing an increase of about 2% in the avg mAP caused by a sharp increase in the strict mAP. Compared to the results on the VL, the model seems to behave well also on the external TS, showing only a slight decrease in the avg mAP. Table 6 shows classification metrics on VL, internal and external TS. On the internal TS, the model still gives good results, improving the F1 score of the VL. On the other hand, the model drastically loses its precision on the external TS while having a high recall, which indicates that the model tended to match the pairs

SBERT	Topic	Pooling	TR	VL			
			Loss	Loss	mAP (S)	mAP (R)	Avg mAP
N	N	cls	0.057	0.226	0.884	0.995	0.939
N	N	avg_last	0.061	0.242	0.868	0.979	0.924
N	N	avg_2ndlast	0.109	0.259	0.869	0.995	0.932
N	N	avg_sumlast4	0.053	0.281	0.881	0.993	0.937
N	Y	cls	0.132	0.205	0.896	0.995	0.945
N	Y	avg_last	0.115	0.212	0.897	0.995	0.946
N	Y	avg_2ndlast	0.043	0.219	0.889	**0.998**	0.943
N	Y	avg_sumlast4	0.049	0.317	0.885	0.991	0.938
Y	N	cls	0.211	0.251	0.789	0.961	0.875
Y	N	avg_last	0.119	0.207	0.877	0.995	0.936
Y	N	avg_2ndlast	0.117	**0.197**	0.872	0.991	0.931
Y	N	avg_sumlast4	0.118	0.249	0.837	0.978	0.907
Y	Y	cls	**0.036**	0.262	**0.901**	**0.998**	**0.949**
Y	Y	avg_last	0.133	0.261	0.882	0.99	0.936
Y	Y	avg_2ndlast	0.052	0.285	0.872	0.995	0.933
Y	Y	avg_sumlast4	0.096	0.236	0.868	0.992	0.93

Table 3: Results of RoBERTa and variants on TR and VL in the second phase of the model selection. SBERT indicates whether SBERT embeddings are concatenated to the transformer outputs before feeding them to the final linear layer. Topic specifies if the topic is being given as input to the model. Pooling indicates the type of pooling employed to get the final contextualized representation from the transformer. (S) = strict, (R) = relaxed, Loss = cross entropy loss. Avg mAP is the average between strict and relaxed mAPs.

	Accuracy	Precision	Recall	F1
TR	0.990	0.978	0.973	0.976
VL	0.907	0.725	0.911	0.807

Table 4: Classification metrics on TR and VL for the best model of the second phase of the model selection. The classification threshold is set to 0.5.

too often. This, together with the model selection results, indicates that the used metrics may favor models which do not perform necessarily well on non ambiguous labeled data.

8 Error analysis

To complete the study, errors on the external TS have been analyzed manually in order to try to understand the weaknesses of the model. To avoid choosing a particular threshold for the classification labeling, only the pairs which the model was most confident about have been considered for the study.

Since the final model has a low precision, most of the analysis focused on the pairs labeled with 0 that have been wrongly matched. What emerged was that, at first glance, many of these pairs seemed to be matching also for a human. For humans, a reasonable match score can be assigned only after seeing all the possible key points and the general distribution of the arguments. To show an example, consider the following topic, argument and key point in order: *"Routine child vaccinations should be mandatory"*, *"Child vaccination should be mandatory to prevent children from spreading the virus"*, *"Routine child vaccinations are necessary to protect others"*. The argument seems to be matching the key point to a high degree. Let us now introduce a key point that actually matches the argument: *"Routine child vaccinations should be mandatory to prevent virus/disease spreading"*. The argument seems to be matching this second key point more than the first one. The score assigment can be also influenced

	Loss	mAP (S)	mAP (R)	Avg mAP
VL	0.262	0.901	0.998	0.949
Internal TS	0.208	0.940	0.996	0.968
External TS	0.366	0.900	0.977	0.939

Table 5: Results of the final model on VL, internal TS and external TS. mAP (with no letters) is computed with labeled data only. (S) = strict, (R) = relaxed, Loss = cross entropy loss. Avg mAP is the average between strict and relaxed mAP.

	Accuracy	Precision	Recall	F1
VL	0.907	0.725	0.911	0.807
Internal TS	0.929	0.796	0.868	0.830
External TS	0.883	0.585	0.933	0.719

Table 6: Classification metrics on VL, internal TS and external TS of the final model. The classification threshold is set to 0.5.

by the nature of the other arguments. If the arguments generally happen to match key points in a strict way, one may end up assigning lower scores to looser but still matching pairs. This indicates that, perhaps, the model should be informed about the existence of other key points and arguments to assign the scores in a reasonable way.

Other more subtle errors are caused by the model wrongly associating different terms coming from the same semantic field. Consider for example the following topic, argument and key point: *"The USA is not a good country to live in"*, *"The USA is a bad country to live in because some people are aggressive"*, *"The US is xenophobic/racist"*. Here, the model is fooled by the semantic similarity between aggressiveness and xenophobia/racism. More specifically, since transformer based models are pretrained focusing only on the co-occurrence of words, the model is not aware of the actual logical relationship between them. Consequently, it can not know that someone who is aggressive is not necessarily xenophobic or racist.

Lastly, some errors depend on the bad quality of the arguments and of the labeling. Consider the following triple: *"The USA is not a good country to live in"*, *"Not because it is a country with many restrictions for immigrants."*, *"The US is xenophobic/racist"*. The model correctly assigns a low score to the input. However, the pair is labeled with a 1, indicating that the annotator interpreted the initial *"Not"* as a mining mistake.

9 Conclusions

Key point analysis is a promising way to summarize a large number of arguments about a certain topic. In this paper, which describes a solution for the Track 1 of the KPA 2021 shared task, the specific problem of assigning match scores to `<argument, key point>` pairs was tackled by framing it as a text classification task, focusing on different state of the art transformer-based architectures and variants specifically tailored to the problem. The analysis included a number of different transformers, namely BERT, XLNet, RoBERTa and ALBERT, as well as sentence embedding transformers, such as SBERT. After a lightweight model selection procedure, RoBERTa managed to beat the other models under the metrics imposed by the competition. Informing the model about the topic (appropriately modified based on the stance) has been beneficial for the overall performance. On the other hand, supplying the model with information about the semantic similarity between arguments and key points through SBERT embeddings gave no particular advantage. Similarly, changing the type of pooling of the transformer's hidden representations did not result in any notable difference. After the evaluation on the external TS, the model maintained a good behavior under the specified metrics, which take into account ambiguous non-labeled data, while getting a low precision on non-ambiguous data, showing that the metrics may favor imprecise models. Lastly, the final error analysis showed how the classification problem as formulated in the paper may be underspecified

and that the model should also be informed about the existence of other key points and arguments. Moreover, particular errors highlighted the intrinsic weaknesses of transformer based models in the task of key point matching.

Many choices and results of this work have been determined by computational limitations. With more computational power, one could improve the model selection by employing a robust cross-validation procedure and by testing other valuable architectures. Finally, as suggested by the error analysis, an interesting direction in the key point matching related research would be to experiment with models that take information regarding multiple key points and arguments when computing a match score for a single <argument, key point> pair.

References

Yamen Ajjour, Milad Alshomary, Henning Wachsmuth, and Benno Stein. 2019. Modeling frames in argumentation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2922–2932, Hong Kong, China. Association for Computational Linguistics.

Roy Bar-Haim, Lilach Eden, Roni Friedman, Yoav Kantor, Dan Lahav, and Noam Slonim. 2020a. From arguments to key points: Towards automatic argument summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4029–4039, Online. Association for Computational Linguistics.

Roy Bar-Haim, Yoav Kantor, Lilach Eden, Roni Friedman, Dan Lahav, and Noam Slonim. 2020b. Quantitative argument summarization and beyond: Cross-domain key point analysis. *arXiv preprint arXiv:2010.05369*.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*, 6(4):1–220.

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

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.

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

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

Amita Misra, Brian Ecker, and Marilyn Walker. 2016. Measuring the similarity of sentential arguments in dialogue. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 276–287, Los Angeles. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Nils Reimers, Benjamin Schiller, Tilman Beck, Johannes Daxenberger, Christian Stab, and Iryna Gurevych. 2019. Classification and clustering of arguments with contextualized word embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 567–578, Florence, Italy. Association for Computational Linguistics.

Lichao Sun, Kazuma Hashimoto, Wenpeng Yin, Akari Asai, Jia Li, Philip Yu, and Caiming Xiong. 2020. Adv-bert: Bert is not robust on misspellings! generating nature adversarial samples on bert. *arXiv preprint arXiv:2003.04985*.

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. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

A Classification results with F1 maximizer classification threshold

	Model	Type	Th	Accuracy	Precision	Recall	F1
TR	SBERT	paraphrase-mpnet-base-v2	0.666	0.749	0.418	0.535	0.470
	BERT	bert-base-uncased	0.459	0.963	0.906	0.918	0.912
		bert-large-uncased	0.443	0.96	0.901	0.904	0.903
	XLNet	xlnet-base-cased	0.498	0.894	0.752	0.731	0.742
		xlnet-large-cased	0.376	0.916	0.791	0.806	0.798
	RoBERTa	roberta-base	0.315	0.948	0.877	0.869	0.873
		roberta-large	0.264	0.983	0.958	0.959	0.958
	ALBERT	albert-base-v2	0.132	0.966	0.92	0.914	0.917
		albert-xxlarge-v1	0.697	0.974	0.95	0.923	0.936
VL	SBERT	paraphrase-mpnet-base-v2	0.666	0.744	0.430	0.619	0.508
	BERT	bert-base-uncased	0.459	0.814	0.542	0.836	0.658
		bert-large-uncased	0.443	0.815	0.544	0.827	0.656
	XLNet	xlnet-base-cased	0.498	0.89	0.768	0.695	0.73
		xlnet-large-cased	0.376	0.851	0.625	0.757	0.685
	RoBERTa	roberta-base	0.315	0.87	0.654	0.829	0.731
		roberta-large	0.264	0.866	0.626	0.927	0.748
	ALBERT	albert-base-v2	0.132	0.847	0.602	0.835	0.7
		albert-xxlarge-v1	0.697	0.929	0.799	0.889	0.842

Table 7: Classification results on TR and VL with F1 score optimizing classification threshold in the first phase of the model selection.