Modern Talking: Key-Point Analysis using Modern Natural Language Processing

Max Henze and **Hanh Luu** and **Jan Heinrich Reimer**
Martin Luther University Halle-Wittenberg, Germany
{max.henze,thi.luu,jan.reimer}@student.uni-halle.de

Abstract

TODO

1 Introduction

Test citation (Bar-Haim et al., 2020)

TODO

2 Related Work

2.1 Key Point Analysis

Given the first track of the shared task analyzing key points is crucial. In their work (Bar-Haim et al., 2020) Bar-Haim et al. propose an approach for summarising large argument collections to small sets of key points. Thus covering a sufficient amount of all arguments. They show that domain experts can very quickly create pro and con key points, which are able to "capture the gist" of the arguments on the given topic. All this without being exposed to the arguments themself. Furthermore they develop the large-scale dataset ArgKP which is the foundation of this shared task. In a later work (Bar-Haim et al., 2020) Bar-Haim et al. construct an automatic method for key point extraction which can compete with key points created by human domain experts. The method consists of two aspects. Assuming that the key points can be found among the given comments they first select short, high quality comments as key point candidates an then select the candidate with the highest data coverage. Using the HuggingFace transformer framework they fine-tune four different models from which ALBERT (Lan et al., 2019) has the best F1 score with 0.809 but RoBERTa (Liu et al., 2019) (F1 score of 0.773) is chosen for key point extraction since it has a 6 times faster inference time. In the paper (Egan et al., 2016), Egan et al. propose a summarising for informal arguments such as they occure in online political debates. By extracting verbs and their syntactic arguments they retrieve points which can make key content accessible. By grouping these points they propose to create discussion summaries.

2.2 Argument Clustering

Argument Clustering is a mighty tool that enables algorithms to assign multiple arguments, which adress a similar key message to a given topic. In the paper (Reimers et al., 2019), Reimers et al. make use of "contextualize word embeddings to classify and cluster topic-dependet arguments". Having performed argument classification they then compute similar and dissimilar pairs of arguments. Two approaches one with clustering and one without are being used. Clustering arguments is achieved by usage of agglomerative hierarchical clustering (Day and Edelsbrunner, 1984). Without clustering a fine-tuned BERT-base-uncased model reached a F1 mean score for similar and dissimilar arguments of 0.7401. Agglomerative hierarchical clustering being a strict partitioning algorithm, results for clustering perform worse by up to 7.64pp (Bert-large F1 mean score: 0.7135). Hence they conclude that "strict partitioning clustering methods introduce a new source of errors". Another approach proposed in a paper (Ajjour et al., 2019) by Ajjour et al. revolves around clustering arguments into so called frames which are "a set of arguments that focus on the same aspect". Thereby framing (Entman, 1993) only a specific information to present to the listeners and convince them of your stance. They propose that an argument consists of two crucial parts. The topic and the frame. Hence their approch splits into three steps: First, all arguments are clustered into m topics. Second, topical features are extracted from all arguments and therefore from its cluster. Third, the arguments are reclustered into k non-overlapping frames. By utilizing k-means (Hartigan and Wong, 1979) for clustering and *Term Frequency-Inverse Document Frequency* (TF-IDF) for topic removal they achieved a F1 score of 0.28.

2.3 Stance Classification

By knowing the stance of an argument it becomes nearly effortless for a human to classify it as pro or con to a given topic.

3 Data

The dataset used in this shared task is the IBM Debater(R) - ArgKP v1 (Bar-Haim et al., 2020) which consists of over 24.000 argument and key point pairs labeled as matching/non-matching. They all belong to one of 28 controversial topics ranging from "Assisted suicide should be a criminal offence." to "We should abandon marriage." therefore drawing a clear line between supporting and non-supporting key points.

The section used for training has over 5500 arguments belonging to over 200 key points with 24 topics. This leaves the dev dataset with over 900 arguments and 36 key points for 4 topics.

3.1 Characteristics

TODO

4 Approach

In this section we present a rule-based baseline performing well, given it's simplicity. In addition to this we came up with two BERT and RoBERTa based approaches. Both improving our baseline.

4.1 Baseline

We start with a very simple baseline. Therefore choosing a Term Overlap baseline with preprocessed terms. Generally it can be assumed, that key-points are summarizing ideas of all associated arguments. We therefore came up with the idea that certain key-words, contained in a lot of arguments, are also very likely to be present in the associated key-point. This makes sense with our intuition, because rather than using completly new words for summarization of arguments, a human would rather reuse certain important words, which have been already found in the arguments.

For example, the following argument exists in the ArgKP dataset:

People reach their limit when it comes to their quality of life and should be able to end their suffering. This can be done with little or no suffering by assistance and the person is able to say good bye.

In relation to this the following key-point can be found:

Assisted suicide reduces suffering.

It can already be seen, that an overlap with the words *suffering* exists. We can further increse this overlap by performing simple preprocessing steps. First of all, we utilize stop word removal for reducing the noise within all arguments. Initially this can be seen counter productive, because less words means less overlap and therefore worse performance. But at second glance this makes a lot of sense. A lot of arguments and key-points contain unnecessary words like *the, and, as etc.*. Removing these gives us purer sentences and results in less confusion with the term-overlap algorithm. Furthermore the redundancy of language makes it possible to contain key-aspects in sentences, even with out these unnecessary stop words.

Secondly, Stemming reduces terms to their corresponding stems and thus achieves a better generalization, when comparing terms. For example, the word: *weakness* will be stemmed to *weak* using the Porter-Stemmer (Porter, 1980). Thus creating an overlap between those words and increasing the possibility that an argmunt containing *weakness* will be associated to a key-pont containing *weak*.

Thirdly, we increase the generalization of our term-overlap algorithm even further by creating lists of synonyms and antonyms and testing if checked words can be replaced with candidates from these to increase overlap.

For the actual similarity computation of given arguments and key-points we use the Jaccard similarity coefficient (Jaccard, 1902). Meaning a higher proportion of terms that appear in an argument as well as in a key-point will classify this key-point to be more likely to match.

5 Results

TODO

6 Conclusion and Future Work

TODO

6.1 Future Work

TODO

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 2915–2925.

Roy Bar-Haim, Lilach Eden, Roni Friedman, Yoav Kantor, Dan Lahav, and Noam Slonim. 2020. From arguments to key points: Towards automatic argument summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4029–4039. Association for Computational Linguistics.

Roy Bar-Haim, Lilach Eden, Roni Friedman, Yoav Kantor, Dan Lahav, and Noam Slonim. 2020. From arguments to key points: Towards automatic argument summarization. *arXiv preprint arXiv:2005.01619*.

Roy Bar-Haim, Yoav Kantor, Lilach Eden, Roni Friedman, Dan Lahav, and Noam Slonim. 2020. Quantitative argument summarization and beyond: Cross-domain key point analysis. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 39–49. Association for Computational Linguistics.

William HE Day and Herbert Edelsbrunner. 1984. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, 1(1):7–24.

Charlie Egan, Advaith Siddharthan, and Adam Wyner. 2016. Summarising the points made in online political debates. In *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*, pages 134–143.

Robert M Entman. 1993. Framing: Toward clarification of a fractured paradigm. *Journal of communication*, 43(4):51–58.

John A Hartigan and Manchek A Wong. 1979. Ak-means clustering algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1):100–108.

Paul Jaccard. 1902. Lois de distribution florale dans la zone alpine. *Bull Soc Vaudoise Sci Nat*, 38:69–130.

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*.

Martin F Porter. 1980. An algorithm for suffix stripping. *Program*.

Nils Reimers, Benjamin Schiller, Tilman Beck, Johannes Daxenberger, Christian Stab, and Iryna Gurevych. 2019. Classification and clustering of arguments with contextualized word embeddings. *arXiv preprint arXiv:1906.09821*.

A Appendix

We release our source code online under the free MIT license.[1]

[1] https://github.com/heinrichreimer/modern-talking