KEY POINTS EXTRACTION METHODS FOR THE LEGAL DOMAIN

Yauheni Mardan*[1], Rayen Dhahri*[1], Oren Sultan*[1,2]

[1]The Technical University of Munich, Germany. [2]The Hebrew University of Jerusalem, Israel.
* These authors equally contributed to this work.

Introduction

Key Point Analysis (KPA) was introduced in BarHaim et al. as a challenging NLP task with close relation to Computational Argumentation, Opinion Analysis, and Summarization. Given a collection of relatively short, opinionated texts focused on a topic of interest, the goal of KPA is to produce a concise list of the most prominent **Key Points (KPs)**. KPA can be used to gain better insights from public opinions expressed through social media, surveys, etc.
This new task was initially suggested with the new **ArgKP 2021 dataset** for the argument-KP mapping task. It has 24,000 (argument, KP) pairs labeled as matching/non matching.
In our work, we focus on the **legal domain** – given a collection of premises from **judgments**, our goal is to extract the most prominent KPs from the premises. We explored three different methods for the KP extraction task. The results of the methods will be evaluated in the matching task.

KP Extraction Task

Given a collection of arguments towards a certain topic, the goal is to generate KP-based summary. KPs should be concise, non-redundant and capture the most important points to the topic of interest. Ideally, they should summarize the input data at the appropriate granularity - should be general enough to match a significant portion of the arguments, yet informative enough to make a useful summary. Then, in the **matching task**, the goal is to compute the confidence score between arguments to the extracted KPs.

Dataset

We use the **European Court of Human Rights (ECHR)** dataset to extract KPs in all of our methods. It consists of 42 human-annotated **judgements**. The corpus is annotated in terms of **premises** and **conclusion**. Overall, 1951 premises and 743 conclusions. We consider the premises as **arguments**, and extract the KPs from them.
Example of an argument:
"The Commission considers that this indicates an issue falling within the scope of freedom of expression.".

Figure 1: The difference between embeddings distributions of our dataset (**left**) and ArgKP 2021(**right**).

Method I : KP Candidate Extraction and Selection Using IBM Debater

We use the KPA of BarHaim et al. (IBM Debater) which extracts KPs in two steps.
- **Candidate Extraction**: find KP candidates from the arguments. When extracting candidates, we assume that KP can be found in the given arguments. We choose to filter sentences with less than **4** tokens or more than **36** tokens.
- **KP Selection**: the most salient candidates are selected as KP. We choose a mapping threshold of **0.9** (a higher threshold leads to higher precision and a lower coverage). We match a sentence to only **one** KP (the one with the highest matching score).

In the output, one argument is broken into multiple sentences (each possibly connected to a different KP). Hence, we concatenate the sentences back to the original argument and choose the KP with the maximum score. We propose two pipelines:
- Create KPs out all the judgment's texts
- **Create KPs for every judgment's text separately and union KPs**

Since judgements are independent, it makes more sense to go with the second pipeline. Indeed, it creates higher-quality KPs.

Method II : Clustering and Summarization

This method consists of two steps: **arguments clusterization** and **summarization of clustered arguments**.
- **Arguments clusterization**: We first encode the arguments to embeddings using **LegalBERT**. Then, we cluster the embeddings of the arguments for each text **separately** using the **HDBSCAN** algorithm.
- **Summarization of clusters**: Now we apply summarization on each cluster of arguments. We suggest two approaches:
  - **Extractive**: we used **LexRank**, **LSA**, **LunH**, and **KL-Sum**. In these methods the resulting KP is one of the original arguments.
  - **Abstractive**: we used **LegalPegasus**. Pegasus models summarize arguments in a new sentence, in contrast to before.

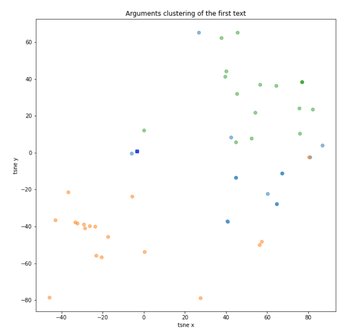

Figure 2: Example of highly clustered arguments. Blue square is the example of the argument shown in the **Dataset** section.

Method III: PageRank and Clustering

We considered two approaches:
- **Quantitative** based: Concatenate the arguments of each judgment's text and perform **Pagerank** as follows:
  - Encode the arguments to embeddings using **LegalBERT**. Then, calculate the cosine similarity matrix between them. The ones with the highest number of comparisons that pass a **minimum threshold** (**0.8**) are KP candidates.
  - From these KP candidates, we take **N** KPs with cosine similarity score below a **maximum threshold** of **0.4** to avoid covering semantically similar KPs. The **N** and the **maximum threshold** are hyperparameters manually tuned based on the number of arguments.
- **Clustering** based:
  - Cluster all arguments of each judgment's text using **HDBSCAN** algorithm.
  - Repeat the same step of the first approach from encoding to the hyperparameter's selection. Here we define the **maximum threshold** and the **N** number of KPs to generate as follows:
    - **maximum threshold** is tuned by using the **distance between clusters**.
    - **N** is set to the number of clusters per judgment text.

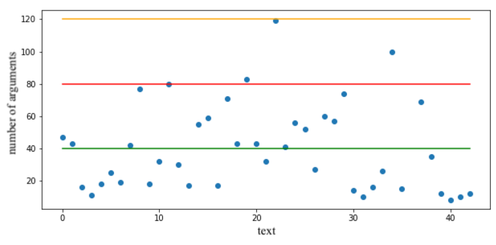

Figure 3: Number of arguments per text. The green, red and yellow lines are the boundaries used to decide the **Number of KPs** to extract

Methods example output

These outputs are KP candidates for the example argument from **Dataset** section.
Method I: Everyone has the right to the freedom of expression.
Method II: The Commission finds no evidence in the case to substantiate this complaint.
Method III: The Commission finds that the applicant was deprived of his liberty after conviction by a competent court within the meaning of Article 5 para.

Conclusion

We tackled the new task of KP extraction, on a new dataset from the **legal domain**. We show three different methods to generate/extract KPs from premises (arguments) of judgment's texts. Notice that, to be able to compare the methods by metrics, they should be evaluated as part of the **matching task**.
Some important notes about the methods:
- In **methods I and III** we create KPs which have already appeared in the arguments, while in **method II** we are able to create KPs which are new sentences.
- In **method I**, there is a tradeoff between **coverage** and the **precision** (How many KPs match to their arguments) of the KPs.
- **Method II** is more **flexible**. It enables us to choose different clustering and summarization algorithms.
- **Method III** allows us to determine the number of KP to extract and their granularity **in advance**, thanks to the filtering pipeline.