TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technologies

Artem Filipenko 214035IVSM

DATA AUGMENTATION TECHNIQUES FOR ADVANCED END TO END KEYPHRASE EXTRACTION FROM TEXT

Master's thesis

Supervisor: Tanel Alumäe, PhD

Tallinn 2023

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Artem Filipenko 214035IVSM

ANDMETE AUGMENTEERIMISE TEHNIKAD VÕTMESÕNADE EKSTRAHEERIMISEKS TEKSTIST TÄISAHELA MEETODIL

Magistritöö

Juhendaja: Tanel Alumäe, PhD

Tallinn 2023

Abstract

This thesis is focused on applying data augmentation approaches to increase performance of the sequence-to-sequence model, trained to extract keyphrases from the text. Keyphrase extraction describes the process of automatically identifying the most important words or phrases in a text document. Other than that, it is an important component of natural language processing. In the scope of the current thesis both extractive (mentioned directly in the text) and abstractive (describing an abstract context) keyphrases are taken into account.

Data augmentation techniques are used to improve the training data, introducing more diversity to the dataset. The goal is to increase performance metrics due to lower overfitting and/or better generalization. The large amount of diverse training data forces the model to handle unseen inputs with a better performance due to a larger number of unusual training samples.

The work presented contains various experiments with a wide spectrum of data augmentation techniques applied, from simple shuffling to more complex approaches, including separate models or algorithmic tools like WordNet to generate unseen data based on the default dataset. The thesis is mainly focused on the Inspec dataset due to its smaller size, allowing a larger number of experiments while saving computational resources and time.

The main focus of this thesis is discovering an approach, giving the best abstract keyphrases extraction performance. This method includes augmenting the data by replacing certain words with their synonyms to get as diverse default dataset multiplications as possible. The relative increase in models' performance after training on the generated dataset reached ~68% for abstractive keyphrases and ~11% for extractive ones, compared to the basic approach.

This thesis is written in English and is 42 pages long, including 7 chapters, 14 figures and 15 tables.

Annotatsioon

Käesolev lõputöö keskendub andmete augmenteerimise lähenemisviiside rakendamisele, et suurendada märksõnade tekstist eraldamise täpsust n-ö otsast-lõpuni närvivõrgupõhises mudelis. Märksõnade eraldamine on tekstidokumendi kõige olulisemate sõnade või fraaside automaatne tuvastamine. Lisaks on see loomuliku keele töötluse oluline komponent. Käesoleva lõputöö raames võetakse arvesse nii ekstraktiivseid (tekstis otseselt mainitud) kui ka abstrahhiivseid (abstraktset konteksti kirjeldavaid) võtmesõnu.

Treeningandmete augmenteerimise tehnikaid kasutatakse masinõppe andmete täiustamiseks, mis muudavad andmestiku mitmekesisemaks. Eesmärk on parandada mudeli täpsust tänu väiksemale ületreenimisele ja/või paremale üldistamisele. Masinõppe erinevate sisendandmete mitmekesisus suunab mudelit käsitlema uusi sisendandmeid parema täpsusega tänu suuremale arvule erinevatele treeningnäidetele.

Esitatud töö sisaldab mitmesuguseid eksperimente erinevate andmete augmenteerimise tehnikatega, alates lihtsast võtmesõnade ümberpaigutamisest kuni keerukamate lähenemisviisideni, sealhulgas eraldi mudelid või algoritmilised tööriistad nagu WordNet, et luua esialgsel eandmekogumil põhinevaid seniesinemata andmeid. Lõputöö keskendub peamiselt Inspeci andmekogumile tänu selle väiksusele, mis võimaldab suuremat arvu eksperimente, säästes samal ajal arvutusressursse ja -aega.

Käesoleva lõputöö põhieesmärk on leida lähenemine, mis annab parima abstraktsete võtmefraaside eraldamise jõudluse. See meetod hõlmab andmete augmenteerimist, asendades teatud sõnad nende sünonüümidega, et saada võimalikult erinevaid treeningandmete kogumeid. Mudelite täpsuse suhteline kasv võrreldes tavapärase lähenemisviisiga oli pärast genereeritud andmekogumiga masinõpet kuni ~68% abstraktsete märksõnade puhul ja kuni ~11% ekstratiivsete märksõnade puhul.

Käesolev lõputöö on kirjutatud inglise keeles ja koosneb 42 leheküljest, sealhulgas 7 peatükist, 14 joonisest ja 15 tabelist.

List of abbreviations and terms

OED	Oxford English Dictionary
EDR	Electronic dictionary
TF	Term frequency
NLP	Natural language processing
ML	Machine Learning
MT	Machine Translation
POS	Part of speech tag
LSTM	Long short-term memory
BERT	Bidirectional Encoder Representations from Transformers
BART	Bidirectional Auto-Regressive Transformers
SLURM	Simple Linux Utility for Resource Management
NLTK	National Language ToolKit

Table of contents

List of figures

List of tables

1 Introduction

1.1 Problem statement

Keyphrase extraction is the process of automatically identifying the most important words or phrases in a text document. It has become an increasingly important research area in natural language processing and information retrieval due to its ability to provide concise summaries and meaningful representations of large amounts of textual data. Keyword extraction is important nowadays in the sphere of text data classification, when it is not possible to manage the huge amount of information manually. The extraction of keyphrases can be valuable in a wide range of applications, such as document classification, information retrieval, automatic summarization and text mining. Precisely, in real life it can be used for articles, news or other complex textual information as even now often this work is done manually by the writer or publisher. The reason for having a keyword list for, let's say, articles is to simplify the navigation through the significant data amount and give an overview of the contents more precisely. According to statistics, 80% of the generated data is unstructured [1], which created noticeable problems with analyzing and processing. So, keyword extraction is an extremely useful base to mark and structure any data without having to waste human working hours for this.

To conclude, the most common places to use such systems are news portals, abstract and citation databases, blog platforms etc. Using the automated systems allows to get rid of the human factor, makes extraction follow the same logic and be consistent regardless of a certain author style and helps writers to save their work time.

The main contribution of the current work is introducing an advanced data augmentation technique to get a significant boost in abstractive performance for keyphrases, not mentioned directly in the text, by applying additional algorithmic NLP approaches to the default training data.

1.2 Research objective

The research objective of this article is to figure out a method to create a keyphrase extraction model capable not just to extract keyphrases from the text, but also abstract meanings and topics not mentioned explicitly, transforming them to semantically correct phrases. Another objective is to propose some methods to improve its performance and test that approach using

the same evaluation dataset as for the starting model. The goals are like that as by understanding the best methods for keyphrase extraction, we can enable more efficient and accurate analysis of large volumes of textual data, ultimately improving the effectiveness of many natural language processing applications.

1.3 Research questions

During this research the main goal is to verify whether the success rate of existing keyword extraction models can be improved. This task implies that to accomplish this a certain approach should be tried and tested, therefore this work requires a practical research. Therefore, the following research questions were formulated:

- RQ1: Which data augmentation technique should be considered as the best one for the abstractive keyphrase extraction task in English?
- RQ2: What are the underlying mechanisms by which the chosen data augmentation approach improves the models' performance?
- RQ3: How big is relative model performance improvement, gained by applying the chosen data augmentation technique?

1.4 Outline

The current work consists of seven chapters. The first one introduces the problem statement, explaining why this topic is important nowadays along with the research goals and contribution. The next, second chapter gives theoretical information and the history of techniques to solve the problem in the academic sphere. Third chapter gives an overview of existing papers and works in the area of the problem, allowing us to compare the techniques applied. The fourth chapter describes the methodology and preparation for the experiments part, described in the fifth chapter. In the experimentation part various techniques are tested and compared, giving also the possible reasons for certain approaches to fail or be successful. Also it proves the main discovered data augmentation technique (synonyms replacement) actually worked as expected by introducing the additional experiment. The sixth chapter proposes some possible future improvements to the latest approach, described in the fifth chapter. The final, seventh chapter makes the final conclusion and gives the most noticeable results overview.

2 Background

2.1 Relevant concepts and theory

The main problem of the task is to choose the right methodics to compare the value of the word/phrase in the sentence along with sentence importance assessment. It's not a secret that the main field for these kinds of tasks is NLP - Natural Language Processing. It involves the development of algorithms and techniques to analyze and process human language. In the context of keyphrase extraction, natural language processing is used to identify the most important words and phrases in a given text. Another involved concept is an information retrieval topic - a part of NLP, which allows to work with large amounts of data and extract valuable information from them to be analyzed by more precise NLP methodics. There are three main approaches to solve this task - Statistical, Linguistic and using Machine Learning [2]. Also, it is possible to combine these methods resulting in the fourth Hybrid method with various possible proportions and methodics used. To choose the right approach it would be good to review how they work and which one is the most suitable for the keyword extraction purpose (see *Figure 1*).

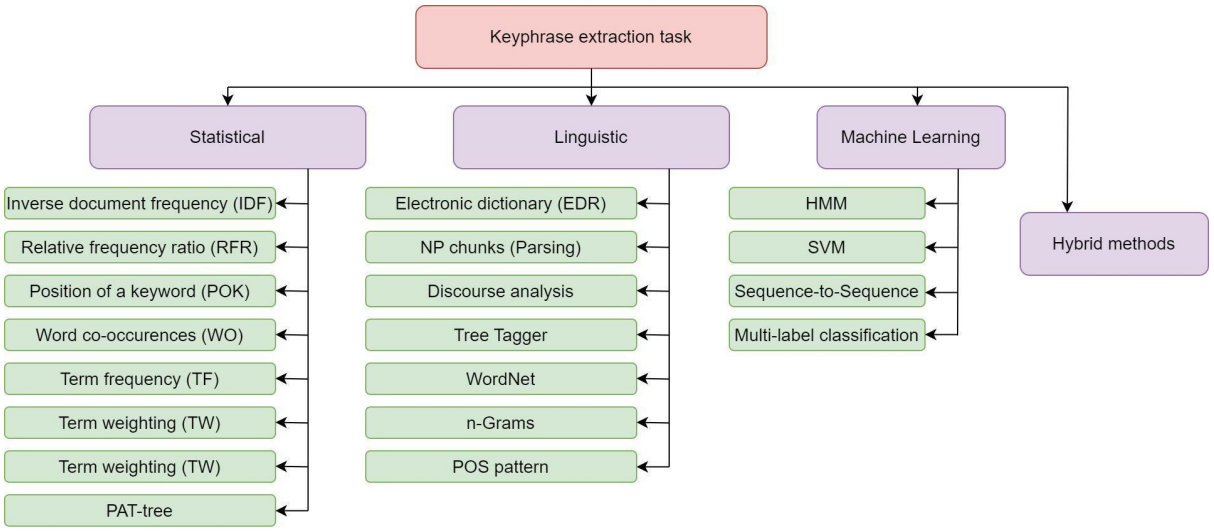

Figure 1. Keyphrase extraction task approaches [41].

2.1.1 Statistical method

Statistical method is the simplest one in the meaning of required resources, however they are not flexible and the algorithm is very predictable and straightforward. The main idea is determining keyphrase importance by analyzing its frequency in the text. The more it is being

mentioned, the more important it is going to be marked. Also phrase length and positions in the text can be taken into account [3]. Some statistical techniques mentioned can be found in Figure 1, however, that's a rare case, when these methods are used alone due to the high error probability and low flexibility. So, often people use combinations of these approaches, such as:

- TF-IDF - terms are sorted by term frequency (TF) in the text and the number of mentions in other documents (IDF) [4].
- TW-IDF - mostly used for retrieving user-related information from the text (however can also be applied for keyphrase extraction), the main difference from TF-IDF is that it takes into account term weight instead of term frequency, which allows to consider not only the number of mentions, but also the importance in the context [5].

Important to mention, that some parts of statistical methods can be applied for data augmentation with the ML approach. For example, it is possible to statistically analyze training data and increase the number of samples with rarely mentioned keyphrases, which can make the model pay more attention to them as well. This can be helpful as usually the training data is highly unbalanced and contains much more majority class samples than minority ones [6].

2.1.2 Linguistic approach

Linguistic approach uses grammar analysis and is dedicated to extract not just words, but the term semantics, which allows to get better results as this system can recognize semantically similar keywords and unify them in the answer [3]. Out of the mentioned methodics in Figure 1 only the combinations of the methodics can give decent results as well as with statistical approach. That becomes obvious after a closer look on some of the mentioned linguistic methods:

- Electronic dictionary (EDR) and WordNet are two possible options to fetch the lexical information about the word (e.g. whether the word is a verb, divide it into the parts etc.).
- Discourse analysis - according to the OED this is a method of analyzing the texts or utterances longer than one sentence, taking into account both their linguistic content and their sociolinguistic context. This can be applied for keyword extraction tasks by comparing the summed scores of the candidates, which are calculated by the term dominance throughout the text [7].

- POS pattern - part of a speech pattern, usually is used to get n-grams needed based on their POS tag pattern.

That's important to mention, that the most interesting for my thesis topic linguistic approach is to use either the WordNet library or EDR to augment the training data with word synonyms, increasing the number of diverse samples to train on. The final decision on which of the two should be chosen has to consider the EDR not being freely available [8].

2.1.3 Machine Learning approach

However, statistical and linguistic approaches used separately are incapable of finding keywords not mentioned directly in the text (e.g. sphere of knowledge etc). For this case neural networks along with machine learning are the main option to use. However, the best results can be achieved only by combining all three approaches. In the case of keyphrase extraction it means taking the most powerful ML approach as a base and applying statistical and linguistic methods for modifying the training data to get the best set to train on.

2.2 Research design

The main challenge while handling cases when the task is to extract abstract keyphrases absent in the input text as they have to be semantically and grammatically correct. Often even if the keyphrase is extractive (present in the input text), it can still be mentioned in a variety of different forms, so it is important to consider them to be the same and avoid self-repetitive ones. This problem forces developers to apply Natural Language Processing techniques based on the text generation to not only extract some information, but to generate it based on the extracted data. This allows the model to distinguish the information about key terms present in the text and the actual list of keyphrases it has to deliver. One of the main candidate approaches is to use pre-trained NLP models for text generation (such as BART) as a base for keyword extraction model training and then launch a learning process using the Sequence-to-Sequence approach [9]. This method allows mapping an input sequence (full text) to a yield of a succession of output sequence (keyphrases list) [10]. Using the proposed methodology the resulting model should be more adapted to construct semantically correct and meaningful phrases to describe the abstract meanings not mentioned directly in the text.

One of the problems is how to order target keyphrases in output sequences for the model training process. Most of the existing approaches just concatenate extractive and abstractive arrays and form a single string using some delimiter (comma, semicolon etc.). The problem

with this approach is that in this case our model can be too precise in the meaning of ordering extractive and abstractive phrases, as in the learning target sample string they are always present as two consecutive subsets (e.g., “ext, ext, ext, abs, abs”). Then the model can lose efficiency trying to follow the same rule for all input texts. One of the possible approaches could be “shuffling” keyphrases in the dataset before training process randomly or using more specific algorithm (e.g., save the order of extractive but shuffle abstractive among them, change the order of extractive to perfectly match the order in the input text etc.). This can make abstractive ones more predictable, as the model won’t try to predict them closer to the end of the generated sequence.

Another possible improvement is closely related to the actual generation process of NLP models. To generate an output sequence, the model predicts a certain number of candidates for the next chunk (or substring) and chooses the one with the highest probability. In case the beam search is applied, the model is going to take the chunk with the highest average probability of this chunks’ possible descendants [11]. The sequence is considered to be ready when the EOS (End of String) symbol has the highest probability. Taking this into account, the generation process can be manually modified to make the model predict the specified number of keyphrases. This can noticeably improve the usability as the approach in the baseline paper allows only to take up to *N* entries out of the generated sequence without possibility to have a larger output set.

2.3 Results validation metric

To validate the performance of the model the best way is to use a metric called F-score. It includes such parameters as the number of predicted keyphrases (*True Positives*), missed ones (*False Negatives*) and predicted incorrectly (*False Positives*). To calculate F-score, *precision* and *recall* should be found as the formula is:

$$F\text{-}score = (2 * Recall * Precision)/(Recall + Precision).$$

- Recall is the ratio of correctly and incorrectly predicted labels. It can be calculated using the formula:

$$Recall = TPcount/(TPcount + FPcount)$$

- Precision is the ratio of correct and missed labels. Formula:

$$Precision = TPcount/(TPcount + FNcount)$$

So, as the performance of the model is an F-score value, which could be quite low even for a decent–performing model, it is possible to have some baseline paper with results that could be outperformed, meaning that the new methodics worked better for the implemented system. This would be the "Applying a Generic Sequence-to-Sequence Model for Simple and Effective Keyphrase Generation" [12] mentioned above, one of the latest theses on this topic. The main purpose of the current research is to figure out if it is possible to improve the results and to describe the methods tried. The success criteria is whether the trained model works better than it used to in the baseline or other peoples' works.

2.4 Possible evaluation data problems

In these kinds of tasks the results evaluation is the most arguable topic as there is no clear "ground truth" (which is quite common for NLP tasks in particular). For the approach chosen (Machine Learning) usually some part of the dataset is considered to be the test/evaluation one, however, as they are written by humans, especially a lot of different humans with diverse way of thinking nobody can guarantee the logic for them was the same and all the samples in the dataset follow the same logic to extract keyphrases. This leads the resulting performance to be relatively small. Below you can review the results of the related thesis paper with the name "Applying a Generic Sequence-to-Sequence Model for Simple and Effective Keyphrase Generation" [12] (only part of the table was extracted into *Table 1*):

	Inspec F@10	**NUS** F@10	**SemEval** F@10	**Kp20K** F@10	**Krapivin** F@10
CopyRNN	33.6	31.7	29.6	25.5	25.2
CorrRNN	-	33.0	32.0	-	27.8
CatSeq	30.0	34.9	30.6	27.3	27.4
CatSeqD	33.3	36.6	**35.2**	29.8	**28.5**
bart-base-kp	35.6	35.3	31.1	30.9	25.8
bart-large-kp	**38.7**	**38.0**	32.3	**31.1**	25.0

Table 1. Example F-scores for extractive keyphrases (taken from [12])

The first row of the table represents the datasets used along with the number of keyphrases taken into account (F@10 means F-score for top 10 predicted keyphrases). The first column are the names of the models trained and evaluated on mentioned datasets. The two models below have been trained and evaluated by paper authors. F-scores here are presented in percentage (the maximum value is 100). As a result, the ⅓ of efficiency using F-score metrics is considered to be the base one for such systems.

The situation is even worse if the abstractive keyphrases need to be extracted (*Table 2*):

	Inspec F@10	**NUS** F@10	**SemEval** F@10	**Kp20K** F@10	**Krapivin** F@10
CopyRNN	5.1	7.8	**4.9**	11.5	11.6
CorrRNN	-	5.9	4.1	-	-
CatSeq	2.8	3.7	2.5	6.0	7.0
CatSeqD	5.2	**8.4**	4.6	**11.7**	12.0
bart-base-kp	4.8	5.6	3.0	6.1	11.2
bart-large-kp	**5.4**	4.4	2.8	6.1	**13.2**

Table 2. Example F-scores for abstract keyphrases (taken from [12])

That is true even despite the fact that the sequence-to-sequence model takes into account the whole input text string on each char generation step, therefore theoretically it should be able to extract some general meanings not mentioned in the text directly. The problem here is that datasets are constructed by humans and the content can be only assumed to be correct, as the human factor and differences in thinking process can influence the resulting set a lot. To make some examples with existing problems on some datasets available I can take the midas/inspec Hugging Face dataset [13]. Sometimes for certain texts it contains a lot of knowledge areas of the topic as abstract keyphrases, while for some others not. In the example below it can be seen how unbalanced this is, especially according to the fact that the input text size is even bigger for the sample with id 1036 (*Table 3*):

Sample ID	Abstract keyphrases
1024	"variable-resolution compression", "hierarchical pyramid spatial relationship", "successive-approximation quantization", "behavioral inconsistency avoidance", "image encoding", "embedded coding", "rate control optimization", "decision problem", "progressive transmission utility functions", "information theoretic measure"
1036	"feedback"

Table 3. Comparison of the number of abstract keyphrases in midas/inspec dataset

Also, the human factor can be evidently seen by some simple mistakes, which cannot influence the resulting model performance (as the model doesn't care if the keyphrase is extractive or not), but indeed influences the evaluation results in case extractive and abstractive F-score are calculated separately. For example, in the following example (*Figure 2*) the obvious extractive keyphrase is considered to be abstractive (despite the fact it is present in the text):

id (int64)	document (sequence)	extractive_keyphrases (sequence)	abstractive_keyphrases (sequence)
1,002	["Selective", "representing", "and", "world-making", "We", "discuss", "the", "thesis", "of", "selective", "representing-the",	["selective representing", "mental representations", "organisms", "realism", "cognitive profiles"]	["world-making", "mind-independent world"]

Figure 2. The human factor errors present in datasets.

Therefore, there are two ways to deal with data imperfection - either try to improve it by applying some filtering before augmentation attempts or just ignore the problem as quite a lot of papers do the same and relative results should be similar for the same actual performance. The second option is preferable, as I would like to get as clean relative results as possible, without artificial boost by solving some evaluation data issues.

3 Related work

Keyphrase extraction can be performed using the various approaches, and using the machine learning approach doesn't automatically guarantee the best result. Sometimes algorithmic preprocessing like constructing a word graph based on the co-occurence relationship can help to reduce the amount of noise in the text, as introduced in [14]. While some researchers are focusing on combining neural models with algorithmic approaches, other ones try to improve the performance of the chosen model type. One of such works [12] is going to be considered a baseline for the current theses, despite the fact the slightly different approach was chosen for extracting the data. Despite the same sequence to sequence approach is used, the proposed solution allows to control the number of keyphrases extracted, which can be both an advantage and the weak point of such method as the number of keyphrases present in the input text is better to be figured out by the model relying on the context of the whole document given. The resulting effect in relative evaluation results is especially visible on the datasets with a large amount of pre-defined keyphrases in the training data (like in Inspec), when the dataset contains a larger amount of phrases, considered to be the most important and representing the text context.

So, although most neural network models heavily depend on the quality of the training and evaluation data, the chosen model type and evaluation approach also matters a lot. Some researchers concentrate on using LSTM models [15] to validate the pre-processed list of candidate keyphrases, others are working on the fully end-to-end models [16], allowing to get the result using only the model pipeline (sequence to sequence type is the best for this purpose). However, when using these approaches the performance of the resulting system heavily depends on the training data quality and overall training process along with the base model and hyperparameters values chosen. Using a single end to end sequence to sequence model gives a list of keyphrases directly, avoiding the step of filtering them out of the candidates list, so the model has to correctly guess the number of keyphrases for the current input text, which is again highly tight with the quality of the training data.

Also, although keyphrase extraction and generation terms are often considered to be the same [17], in the machine learning field it makes a huge difference when coming to a decision on which base model is the best to use. Extraction is mainly used to find related features in the input text, so in the scope of this thesis this approach is mainly suitable for outlining extractive keyphrases. For these approaches extractive base models like BERT [18] are often

the best choice. Generation should be less focused on the features in the input text, taking into account the abstract context of the whole document, so it is more suitable to outline abstractive features with BART [19] as a base model.

4 Methodology

Before starting experimenting, a couple of things should be clarified. The first problem is to choose the Python package for training/evaluation. The main requirement for it is to be as configurable as possible and allow to add a custom behavior for data manipulation (including some preprocessing before tokenization), custom evaluation function and easily configurable hyperparameters. After the step above is finished, the next one would be to choose some datasets to use as a base, as this would influence the variety of text topics covered along with GPU computing time and the VRAM size needed for a certain number of batches. The next step is to choose some general language model as a base one to fine-tune on (or move to doing everything from the scratch). The main reason to consider using a pretrained model as a base is to increase the models' dictionary and capability to work with English text data better [20]. If everything is ready, the main experimentation part can start, which includes generating different datasets, data preprocessing or rearranging, changing the hyperparameters value etc.

For training purposes the SLURM workload manager of the UT computational cluster will be used. The configuration includes using 4 CPU cores, 32GB RAM and either Tesla with 32GB VRAM (suitable when training on smaller datasets like Inspec) or A100 with 40GB VRAM (the best choice for larger Kp20k dataset) as a GPU for CUDA computing.

For theoretical questions while preparing and doing the experimentation part the advanced AI-based tools were used to simplify the theoretical research (such as ChatGPT as an advanced Google Search), as well as Grammarly and Microsoft Word as a writing aid when writing the thesis.

4.1 Choice of the base package

As the main goal is to simplify the process as much as it is possible, the best option to use is to join the Hugging Face ecosystem, which is an excellent aggregator for datasets, models and Python libraries used in scripts.

Hugging Face Transformers is a Python package that makes it easy to work with pre-trained models for Natural Language Processing (NLP) tasks like language modeling, text classification, and question answering. The package is built on top of PyTorch and TensorFlow and offers a range of pre-trained models like BERT [18] or BART [19] along

with their variations, which are the most interesting to me within the keyphrase extraction topic.

One of the best things about the Hugging Face Transformers package is that it's easy to use. You can load pre-trained models, process text inputs, and generate outputs with just a few lines of code. For example, you can load a pre-trained BERT model and generate embeddings (in this certain case a set of possible sentence endings) for a sentence like "In theory, I can generate a custom sample here…" by using the AutoTokenizer and AutoModel classes from the package, which can automatically find suitable implementations for your particular model.

You can also fine-tune pre-trained models on custom tasks like text classification or named entity recognition using the package's simple API. This means that you can quickly adapt pre-trained models to your specific needs without having to train them from scratch.

As the base for the script I'll be working on, the best choice is to use the *run_summarization.py* from examples folder. The topic is quite close, as keyphrase extraction is a subtask of summarization, and more importantly it shows the process of sequence-to-sequence model training, including where data can be preprocessed, tokenized and where the results can be evaluated after training.

4.2 Choice of the dataset

Not many datasets are available for keyphrase extraction topics. The main five worth mentioning are KP20k [21], OpenKP [22], KPTimes [23], Krapivin [24] and Inspec [13]. To choose the best dataset to proceed with I would review them and make a conclusion about each:

- KP20k: This dataset contains 532,422 English scientific articles along with their associated keyphrases. The dataset is large and diverse, covering a wide range of scientific fields. However, the keyphrases are sometimes incomplete or contain errors.
- OpenKP: This is a collection of open-access papers from various scientific fields, along with their keyphrases. The dataset is smaller than KP20k, but the keyphrases are generally more accurate. One downside is that the dataset only covers scientific articles.

- KPTimes: This dataset contains articles from the New York Times, along with their associated keyphrases. The dataset is relatively small, but the keyphrases are high-quality and cover a diverse range of topics.
- Krapivin: This is a collection of scientific articles and their keyphrases from various fields. The dataset is relatively small, but the keyphrases are generally high-quality and complete.
- Inspec: This dataset contains abstracts from scientific articles in the field of physics, computer science, and engineering. The dataset is relatively small - only 2000 samples, but the keyphrases extracted are quite decent (except the minor problems I described in the Chapter 2.4) and the dataset covers a diverse range of topics for such a small size.

After reviewing possible options I proceed with two border options - KP20k and Inspec dataset.

Important to mention, that the needed data can be also obtained manually in a custom dataset either by extracting it raw from some data aggregator with subsequent formatting, by combining existing datasets or by generating data using previously trained models. To create a new dataset, the following steps can be followed:

1. Find some data to make the future model find similar patterns and logic in it.
2. Split this data into two or more datasets - usually *training* and *test* (*evaluation*), sometimes also the *validation* one can be added.
    - *Training set:* This is the subset of the dataset that is used to train the machine learning model. The model learns the patterns and relationships in the training set data, and then uses this knowledge to make predictions on new data.
    - *Test set:* This is the subset of the dataset that is used to evaluate the final performance of the model after it has been trained and tuned on the training and validation sets. The test set should be representative of the real-world data that the model will be used on, and it should not be used during training or hyperparameter tuning.
    - *Validation set:* This is the subset of the dataset that is used to tune the hyperparameters of the model. Hyperparameters are settings of the model that are not learned during training, such as the learning rate, regularization strength, or number of hidden layers. The validation set helps to assess how

well the model generalizes to new data and to find the best hyperparameters that minimize the error.

3. Save the dataset as a JSON file, containing the list of samples. If needed, the resulting dataset can be loaded to the Hugging Face dataset repository and accessed there by anyone.

4.3 Choice of the base model

To improve the results, it is recommended to use some pre-trained model to fine-tune. There are two main models, which are mostly used for that purpose - BERT [18] or BART [19]. Their performance can vary depending on the certain task, but in the case of keyphrase extraction an answer to this question is complicated.

BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained transformer-based language model that uses a masked language modeling task to learn contextualized word embeddings. BERT is trained on a large corpus of text and can be fine-tuned on specific NLP tasks by adding a task-specific layer on top of the pre-trained model. BERT has achieved state-of-the-art performance in various NLP tasks, including question answering, sentiment analysis, and named entity recognition.

On the other hand, BART (Bidirectional and Auto-Regressive Transformers) is a pre-trained transformer-based language model that is designed to handle sequence-to-sequence tasks such as text summarization, machine translation, and text generation. BART is trained on a large corpus of text using both bidirectional and auto-regressive modeling objectives. BART has achieved state-of-the-art performance in various sequence-to-sequence tasks, including summarization and machine translation.

An important criteria to decide on which one of these models should be used is whether the information needed is directly mentioned in the text or not. As in the current scope both extractive and abstractive keyphrases have to be extracted, the best way is to proceed with BART. This is because BART has been specifically designed to handle sequence-to-sequence tasks such as text generation and summarization.

To extract abstractive keyphrases using BART, we can use an encoder-decoder architecture that takes the input text and generates the keyphrases as an output. We can train the model on a corpus of text with annotated abstractive keyphrases using a sequence-to-sequence approach, where the model is trained to generate keyphrases that are not explicitly mentioned

in the input text. During inference, the model can generate a list of relevant keyphrases that are not mentioned in the input text but are related to the content.

In conclusion, while BERT is better suited for keyphrase extraction when the task involves identifying relevant keyphrases from the input text, BART is better suited for the extraction of abstractive keyphrases that are not explicitly mentioned in the input text. This is because BART has been specifically designed to handle sequence-to-sequence tasks such as text generation and summarization, which are necessary for generating abstractive keyphrases as in that case the model has to construct meaningful phrases from the scratch.

There are multiple variations of BART-like models (see *Table 4* below), so it is crucial to decide which one to use. In the case of using the midas/kp20k dataset with the highest variety of covered topics, the most logical way is to use a widely specialized base model, such as facebook/bart-large from Hugging Face [25]. However, the same base model can be used for midas/inspec dataset as well, as despite it covers quite a number of topics, they are highly specific scientific ones, which makes them require a broader model knowledge. Still, there is an alternative for the mentioned large model - facebook/bart-base, introduced under the same paper [25], which can be tried as well. The main difference between facebook/bart-base and facebook/bart-large is the size of the model and the number of parameters.

	Description	Params number
bart-base	*bart* model - 6 encoder + 6 decoder layers	135M
bart-large	*bart* model -12 encoder + 12 decoder layers	406M
bart-large-mnli	*bart-large* fine-tuned on MNLI dataset (Multi-Genre Natural Language Inference)	406M
bart-large-cnn	*bart-large* fine-tuned on CNN Daily Mail dataset	406M
bart-large-xsum	*bart-large* fine-tuned on Extreme Summarization XSum dataset	406M

Table 4. Diversity of BART models (data is taken from [25])

The base one has 12 layers of transformers with a hidden size of 768 and 135 million parameters, while the large one has 24 layers of transformers with a hidden size of 1024 and 406 million parameters. Also it would be important to mention, that while base models

bart-base and *bart-large* were trained using self-supervised objective, the fine-tuned versions are trained in supervised way. As a result, BART-large is a larger and more powerful model that can handle more complex tasks than BART-base. To ensure stability and avoid overfitting the best way is to go with the most powerful default variant - *facebook/bart-large*.

4.4 Data preprocessing

The first thing to consider here is the format of existing datasets. For instance, the generation subset of *midas/kp20k* dataset consists of three columns:

- The **document** is the input text, saved as an array of words.
  Example: ["The", "self-organizing", "map", … ,"article."].
- Column **extractive_keyphrases** represents keyphrases directly mentioned in the text. The format is an array of phrases.
  Example: ["self-organizing map", "learning vector quantization"].
- Column **abstractive_keyphrases** represents keyphrases, which describe contents, but are not mentioned in the text directly. The format is an array of phrases.

*Note: only the **document** array is guaranteed to be a non-empty array. Both **extractive_keyphrases** and **abstractive_keyphrases** can be an empty array.*

Now, when the data structure is known, the main preprocessing can start. The chosen model type is Sequence-to-Sequence, which means it requires a tokenized text as an input, and produces some tokenized text as an output. Therefore, the first step is to concatenate the input document into a single string, along with preparing the output single string with keyphrases, separated by some delimiter (“, “ as the most default option). The main complication here is that on the evaluation step it will be necessary to distinguish the predicted extractive and abstractive keyphrases, as the expected output is just a string with phrases, listed with comma as a separator. To fix that, two possible approaches can be used:

The first option is to delegate the classification to the model by adding two special tokens to the training target strings, defining the type of keyphrases after it. Let them be “EXT: “ for extractive keyphrases and “ABS: ” for abstractive ones (*Figure 3*).

Extractive keyphrases
dimensionality reduction
manufacturing
quantization

Abstractive keyphrases
fpga-based hardware emulators
p system
resampling

```
Target string: "EXT: dimensionality reduction, manufacturing, quantization
ABS: fpga-based hardware emulators, p system, resampling"
```

Figure 3. Example target string with both extractive and abstractive keyphrases

In case of using both tokens there is also an option to store the target string for cases, when either extractive or abstractive keyphrases are absent (*Figure 4*):

Abstractive keyphrases
dynamic compilation
embedded system

```
Target string: "ABS: dynamic compilation, embedded system"
```

Figure 4. Example target string with only abstractive keyphrases

The alternative approach is more complicated on the evaluation step, but is reliable and logical in the distinction of extractive and abstractive keyphrases. In this case any special tokens can be skipped, except keyphrase delimiter “, “, resulting in the simpler string with keyphrases, listed with commas. On the evaluation step, however, the system should be implemented to effectively search keyphrase entries over the text. The criteria for the keyphrase to be extractive is whether it can be found in the text. However, there are cases in the text, when some entries are interrupted with punctuation marks or extra spaces, added accidentally on the text concatenation step. Detailed explanation continues on the next subchapter.

4.5 Future evaluation problems

To explain the last problem more precisely, it is necessary to remind, that the document column in any dataset contains each text as an array of separate words. Therefore, to get a single string these words have to be concatenated with a space sign as a delimiter. This is the

easiest option to proceed with, as the model doesn't care whether input data contains any extra space or punctuation mark, only output should be properly formatted to ensure the correct answer format. However, this can lead to certain problems when looking for a specific keyphrase entry in the text, as in some datasets apostrophes can lead to the word being considered as two separate words, splitted by this sign (see *Figure 5* for an example).

id (int64)	document (sequence)	extractive_keyphrases (sequence)	abstractive_keyphrases (sequence)
1,003	["Lob", "'s", "theorem", "as", "a", "limitation", "on", "mechanism", "We", "argue", "that", "Lob", "'s", "Theorem",	["limitation on mechanism", "epistemic authority", "formal system"]	["lob theorem", "belief-set", "theorem-set", "human-like epistemic agents"]

Figure 5. Example of formatting problem for midas/inspec dataset [13]

To get around this problem, both the document and the keyphrase can be converted to searchable format by removing all characters except letters (example is in the *Table 5*). This way the problem is solved in the most optimized way and any possible formatting issue is covered, ensuring that extractive keyphrases in the model output are distinguished correctly.

Before transformation	After transformation
Cooperative three - and four-player quantum games A cooperative multi-player quantum game played by 3 and 4 players has been studied . A quantum superposed operator is introduced in this work which solves the non-zero sum difficulty in previous treatments . The role of quantum entanglement of the initial state is discussed in detail	cooperativethreeandfourplayerquantumgame sacooperativemultiplayerquantumgameplaye dby3and4playershasbeenstudiedaquantumsu perposedoperatorisintroducedinthisworkwhi chsolvesthenonzerosumdifficultyinprevioust reatmentstheroleofquantumentanglementoft heinitialstateisdiscussedindetail
Extractive keyphrases of the text	
quantum entanglement initial state quantum superposed operator nonzero sum difficulty	quantumentanglement initialstate quantumsuperposedoperator nonzerosumdifficulty

Table 5. Example inconsistency of midas/inspec sample (train split, id 114)

5 Experiments

The mainly used ways to improve the model performance can vary depending on the specific problem, type of the model and specific characteristics of the training data. Out of them the following ones are mostly used:

- Modifying of the training data, solving its main imperfections by applying various techniques, mainly known as feature engineering [26]. Here are some main approaches to do that:
    - Scaling - a process of transforming the input data to some consistent range of values. The goal is to prevent some features from being dominant over others, which in some cases can lead the model to be overfitted to recognize certain features (keyphrases in the current case).
    - Normalization - similar process to scaling, with the main difference in the goal of transformation. While scaling adjusts values to a consistent range, normalization transforms input features to make them comparable, without changing relative relationships among them.
    - One-hot (also known as label) encoding allows us to represent categorical input values as numerical data. The general idea is to create a new binary column for each possible state of the categorical number (if the number of categories is limited).
- Hyperparameter tuning, which involves picking the optimal set of hyperparameters to improve the models' performance [27]. The following hyperparameters can be tuned:
    - Learning rate - represents the size of the step to update the weights of the model during training. If the value is too small, the training process can take a long time to reach an optimal state. On the other hand, with large learning rate values the model can fail to achieve appropriate results at all.
    - Dropout rate - the probability of each neuron being randomly dropped out (set to zero). The main goal to use it is to prevent overfitting on a large amount of non-normalized data by resetting individual neurons from being too dependent on each other. Can be useful on a large number of epochs over the same dataset, allowing the model to concentrate on the minor features.
    - Batch size - a hyperparameter, which controls the number of samples processed simultaneously at each iteration. A smaller batch size leads the

model to be updated more frequently with smaller amounts of data, resulting in faster convergence and better generalization, however increasing the noise in estimates. Larger sizes provide better training timings, however can lead to overfitting and poor generalization. The maximum number of samples depends on the sample size and GPU memory used. Also there is a possibility to emulate the higher value of batch size with limited GPU memory by increasing the gradient accumulation, leading to the model being updated only after N subsequent smaller batches are processed.

    - Regularization strength - used to prevent overfitting by adding a penalty term to the loss function while training [28]. This forces the model to have smaller weights (in other words, to be simpler). The smaller value in this case leads to an overcomplicated model with more overfitting, while a higher one allows to get a simpler model. Important to note, that simpler model doesn't mean it will operate better over the validation subset, so increasing the value doesn't guarantee the better result.

- Transfer learning means using some pre-trained model as a base to fine-tune [42], which leads to improvement in performance in most of the cases. It is especially useful in the case of limited training data available (in the case of the current work the midas/inspec dataset can be considered to be the case).
- Data augmentation [29] (which was chosen as the main tested approach of the current paper) involves creating new training data by applying transformations to existing data or generating new one using some text generation model, fine-tuned to generate data backwards (in the case of keyword extraction, the model to generate texts from keyphrases can be implemented).
- Batch normalization is used to balance the inputs to each layer in the model to improve its stability and training speed. On the other hand, it has been reported that such kinds of normalization can increase adversarial vulnerability of the model [30].

In the current paper the main research target is to test data augmentation techniques mainly, however often the most significant boost can be obtained by combining several approaches. To choose the appropriate set of model performance boost methods, various factors have to be taken into account, including the nature of the data, the complexity of the model, the size of the dataset, and the computational resources available.

Out of the approaches listed above, only feature engineering can be excluded as it refers to modifying the initial training data and can interfere with data augmentation techniques. So it can be relatively safely excluded out of the scope to avoid influencing the performance measurement.

One of the approaches, which is going to be used for sure, is transfer learning, as the BART-large model is used as a base for fine-tuning. Using a base model, trained on a huge amount of unlabeled data can give much better results when fine-tuned with a relatively small amount of labeled samples [31].

Another method which can be used in parallel is hyperparameters modification. This technique is especially useful with increased sizes of the training data as a result of data augmentation, as in this case there is a high probability of overfitting. Generated or modified data still has relatively the same structure as an initial training one, so it is important to make sure the model considers as many newly constructed features from the generated dataset as possible, ignoring repetitive ones to hold a balance between them. This issue can be handled by a dropout rate modification, but also the possibility to control model weights update frequency can be useful, for which learning weight and batch size can be modified.

5.1 Experiment 1 - Basic approach

On this step only the basic data preprocessing has to be implemented along with the basic evaluation function to calculate F1 score. The only operation performed before data tokenization is keyphrase concatenation into a single string. Hyperparameters used are either default or taken from the paper considered to be a baseline [12]. The number of epochs is considered to be three, if not explicitly stated otherwise.

The following two datasets are going to be used for this experiment: *midas/kp20k* and *midas/inspec*. The first one was chosen for its complexity and variety of topics, while Inspec is an example of a smaller, but fine-grained dataset.

To make a better overview of an impact, the new type of metrics absent in the baseline paper is going to be added - F@O, which takes the whole set of predicted keyphrases instead of top five for F@5 or top ten in the case of F@10. This became possible because of the approach difference with the baseline, as instead of several ranked sequences the concatenated string with the whole set of predicted keyphrases is produced by the model. The extractive results are listed in the *Table 6* below, along with the abstractive ones in *Table 7*.

	Inspec			Kp20K		
	F@5	F@10	F@O	F@5	F@10	F@O
CopyRNN [32]	29.2	33.6	-	32.0	29.6	-
CatSeq [33]	29.0	30.0	-	31.4	27.3	-
CatSeqD [33]	27.6	33.3	-	**34.8**	29.8	-
bart-base-kp	33.1	35.6	-	32.8	30.9	-
bart-large-kp	35.2	38.7	-	33.1	**31.1**	-
The basic approach (mine)	**44.3**	**48.5**	**48.7**	24.4	24.4	24.5

Table 6. Extractive F1 score comparison with the baseline

	Inspec			Kp20K		
	F@5	F@10	F@O	F@5	F@10	F@O
CopyRNN [32]	-	5.1	-	-	11.5	-
CatSeq [33]	-	2.8	-	-	6.0	-
CatSeqD [33]	-	5.2	-	-	**11.7**	-
bart-base-kp	-	4.8	-	-	6.1	-
bart-large-kp	-	5.4	-	-	6.1	-
The basic approach (mine)	**13.1**	**13.7**	**13.5**	4.8	4.9	4.5

Table 7. Abstractive F1 score comparison with the baseline

The first conclusion can be already made by looking at the results. While performance of the model, trained on the *kp20k* dataset is poor as expected (as no techniques were applied yet), the results for Inspec dataset surprisingly exceed the metrics reported in the baseline. The possible reason is not connected with the evaluation function implementation, as *kp20k* results are the same as expected. Looking at some other models, trained or fine-tuned using *midas/inspec* dataset it can be clearly seen, that the results reported in the baseline for this dataset are surprisingly bad. Self-reported F1 Seqeval (both extractive and abstractive keyphrases) score for *ml6team/keyphrase-extraction-distilbert-inspec* [35] (fine-tuned on the *distilbert* [36], modification of BERT) is 50.9, exceeding the basic approach result. According to the initial assumption, described in the *4.3 Choice of the base model* chapter, BART can be expected to be more suitable for keyphrase extraction tasks with abstractive entries, so the results, reported by [12] when using *midas/inspec* don't seem to be valid enough.

5.2 Experiment 2 - Shuffling as a basic data augmentation

As the training target data in our case is a single string (keyphrases, splitted by commas), the most obvious problem is that during the training process the model can pay too much attention to certain positions of keyphrases in the target string. In theory, this can lead to the model producing the wrong keyphrase just because it was often present at the same position in a significant part of the training data. The first approach tried is simple - instead of doing the N number of epochs over the same target data, it can be re-shuffled before each epoch. As on some frameworks this task can be too complicated and to avoid re-tokenization in the process of training, it is also possible to perform only one epoch over the data, multiplied N times. This method is less effective in terms of RAM usage, but is much faster and simpler in implementation. This way it is possible to get N copies of the data, with target keyphrase strings reshuffled on each virtual epoch. The results are listed in *Table 8* (extractive) and *Table 9* (abstractive).

	Inspec			Kp20K		
	F@5	F@10	F@O	F@5	F@10	F@O
bart-large-kp	35.2	38.7	-	**33.1**	31.1	-
The basic approach	44.3	48.5	48.7	24.4	24.4	24.5
Basic shuffling (current)	**45.1**	**50.8**	**49.4**	31.6	**31.8**	**31.4**

Table 8. Extractive F1 scores for basic shuffling approach (3 epochs)

	Inspec			Kp20K		
	F@5	F@10	F@O	F@5	F@10	F@O
bart-large-kp	-	5.4	-	-	6.1	-
The basic approach	13.1	13.7	13.5	4.8	4.9	4.5
Basic shuffling (current)	**21.1**	**23.7**	**23.5**	**9.8**	**10.0**	**9.8**

Table 9. Abstractive F1 scores for basic shuffling approach (3 epochs)

Even though the basic approach results were incomparable with the baseline, the improvement when using even the simplest shuffling is obvious. The improvement is especially clear for abstractive keyphrases, where the model performs twice as better for both Inspec and Kp20K datasets. The possible explanation is that for abstractive keyphrases it is crucial to let the model understand the position of this keyphrase in the target training sequence is not connected with a certain part of the input text, but only with the whole text as it is.

Another interesting question is how related is the number of epochs to the resulting model performance. To test this, the Inspec dataset is the best choice, as every training iteration for Kp20K can take up to several days for a large number of epochs, while Inspec is compact enough to fit in hours. The results for Inspec dataset are below (*Table 10*):

	Inspec (extractive)			Inspec (abstractive)		
	F@5	F@10	F@O	F@5	F@10	F@O
Basic shuffling - 3 epochs	**45.1**	50.8	49.4	**21.1**	**23.7**	23.5
Basic shuffling - 6 epochs	44.7	51.9	51.6	20.7	**23.7**	**23.7**
Basic shuffling - 12 epochs	44.9	**53.2**	**53.9**	19.8	22.4	22.7

Table 10. F1 scores for basic shuffling approach on various epochs amount

The conclusion is that increasing the number of epochs does work for extractive keyphrases, however leads to a slight decrease in the abstractive extraction performance.

However, this was only the basic shuffling, and the approach seems to be perspective enough, so now some advanced shuffling techniques can be tried.

5.3 Experiment 3 - Advanced shuffling techniques

Shuffling of the keyphrases in target training sequences allowed to improve the evaluation results significantly (especially for abstractive keyphrases), so experiments can be continued. The following algorithm modifications can be tried:

- Order extractive keyphrases in the target sequence by their occurrence in the source text and insert abstractive ones randomly among them.
- Order keyphrases in target sequences by their global occurrence over the dataset train split.
- Apply shuffling to not only the target sequence with keyphrases, but also to the source text (e.g. shuffle sentences).

The experimentation results are as following (*Table 11*):

	Inspec (extractive)			Inspec (abstractive)		
	F@5	F@10	F@O	F@5	F@10	F@O
Basic shuffling	45.1	50.8	49.4	**21.1**	**23.7**	**23.5**
Occurrence in the text	**46.1**	52.7	**52.9**	18.2	23.2	23.1
Global occurrence	43.7	49.9	50.0	17.4	22.0	21.9
Shuffled sentences	45.5	**53.3**	51.6	19.6	22.2	22.0

Table 11. F1 scores for proposed advanced shuffling methods

From the table it can be clearly seen that there are two techniques, providing best performance - ordering extractive keyphrases by occurrence in the source text and shuffling sentences of input texts. For further experiments these two are the best option to apply over any other data augmentation technique, as they can be combined with most of the other possible approaches.

5.4 Experiment 4 - Generating new training data (back-translation inspired)

There is one more approach, often used in MT - using the back-translation based data augmentation. The idea is to involve a separately trained backwards model, trained to generate text from the given list of keyphrases. This idea comes from the human double-checking techniques in math, when something can be verified by applying reverse logic and going from the result to given conditions [34]. The same facebook/bart-large model can be used as a base to fine-tune. The only difference from the training process of the default keyphrase extraction model is that texts are given as model inputs and keyphrase lists as targets in the training data (of course, evaluation step in the end can be skipped, as there is no explicit criteria for such a model to be considered as well-performing). Then, this model can be used to generate any amount of the new training data, using the capabilities of the basic language model used.

The possible profit is the possibility to get a large amount of various training data, increasing the number of diverse samples, which, in theory, would provide a noticeable performance boost compared to training on the same data for several epochs due to lower overfitting chance.

The dataset, used for such generation will be midas/inspec (due to the significantly lower size, compared to *kp20k*). As the default dataset size is 2000 samples, the optimal amount of the new data to be generated is 10000 samples more, to get an analogue of 6 epochs after dataset concatenation.

To control the number of keyphrases, given as an input to the model, this amount can be either randomly generated in a certain range for each sample or preserved the same as in the basic dataset. The first approach should be better, as it provides better sustainability and keyphrase number balance will be preserved the same, as in the base dataset. The easiest way to do that is to save keyphrase amounts of the basic dataset to a list, and then use them in the generation process.

Hugging Face tools allow you to use the datasets as a json file in various formats. In the current work it is going to be the list of Python dictionaries (each is a separate sample).

Sample (data unit)	
extractive_keyphrases	Example structure: [*keyphrase1*, *keyphrase2*, ..., *keyphraseN*]
abstractive_keyphrases	Example structure: [*keyphrase1*, *keyphrase2*, ..., *keyphraseN*]
document	Example structure: [word*1*, *word2*, ..., *wordN*]

Figure 6. The structure of a single sample, which is a Python dictionary

To implement such a system, the following steps should be done:

1. Train the backwards model to generate texts out of lists of keyphrases.
2. Obtain the list of unique keyphrases for the whole Inspec dataset.
3. Save the number of keyphrases for each sample to the separate list.
4. Form the list of batches to give as an input to the backwards model, trained on the first step. To preserve the balance of keyphrase amounts the same as in the basic

model, the lengths from the third step should be taken successively out of the formed circle-closed array.

5. Tokenize resulting batches and pass them to the model input.
6. Decode resulting texts.
7. Split the input keyphrases into two categories (extractive or abstractive), based on their occurrence in the generated text.
8. Save the resulting document, extractive and abstractive keyphrases into the output json file.


```
[
    {
        "extractive_keyphrases":
            ["tumor dose", "flexible polyurethanes", "segmental status", "receiver
            sensitivities", "multivariate polynomial", "optical shadow-casting logic system",
            "layered manufacturing processes", "variable-order formulas", "invertibility",
            "boolean implication", "slightly lossy compression algorithm", "multivariate
            discordant observations"],
        "abstractive_keyphrases":
            ["state legislation", "brain architecture", "brain region truncation", "mammogram
            synthesis", "mesh simplification", "san francisco bay community", "integration"],
        "document":
            ["In", "this", "paper,", "we", "propose", "a", "slightly", "lossy", "compression",
            "algorithm", "to", "reduce", "the", "number", "of", "variable-order", "formulas",
            "to", "a", "single", "multivariate", "polynomial.", "The", "algorithm", "is", "based",
            "on", "an", "optical", "shadow-casting", "logic", "system", "-LRB-", "OSS",
            "-RRB-,", "which", "is", "used", "in", "layered", "manufacturing", "processes",
            "such", "as", "flexible", "polyurethanes.", "In", "addition,", "we", "show", "that",
            "the", "invertibility", "of", "the", "multivariate", "discordant", "observations", "is",
            "invertible", "under", "the", "assumption", "of", "boolean", "implication.", "We",
            "also", "present", "a", "new", "algorithm", "for", "segmental", "status", "of", "the",
            "brain", "region", "truncated", "in", "the", "mammogram,", "which", "can", "be",
            "used", "to", "determine", "the", "tumor", "dose,", "tumor", "architecture,", "and",
            "receiver", "sensitivities"]
    },
    ...
]
```


Figure 7. Example generated data sample

As a result, the new dataset with 10000 samples is saved. From the human point of view, texts don't have much sense, however it should now matter for the process of extractive keyphrases detection. On the other hand, it can become a problem for abstractive ones, as the background meanings of the resulting document can conflict with a given list of abstractive keyphrases.

The only way to validate the model performance after training on this new dataset is to make a test. After concatenation with the default Inspec dataset the number of samples became 12, which is an equivalent of 6 epochs for the default dataset, so there is no need to have more than 1-2 epochs. The results can be seen in the *Table 12* below:

	Inspec (extractive)			**Inspec (abstractive)**		
	F@5	F@10	F@O	F@5	F@10	F@O
Shuffled sentences (previous best)	**45.5**	**53.3**	**51.6**	**19.6**	**22.2**	**22.0**
Generated dataset only	37.1	43.8	42.3	15.7	16.6	16.5
Generated dataset + 6x default Inspec	44.6	52.5	51.8	19.3	21.8	21.6

Table 12. Comparison of using the newly generated dataset with default one

From the results it can be seen that the approach didn't work as expected and can be discarded. The possible reason for that is the evaluation subset of the default Inspec is being used, and the context in generated documents can be changed compared to the default dataset. However, despite the doubts, the performance for abstractive keyphrases is not as awful as expected, which means the backwards model was able to catch the abstractive context and put it into the generated documents.

5.5 Experiment 5 - Round-trip translation approach

As the previous approach (generation of new samples) hasn't given the desired result, it would be better to stick to the context of the original dataset without changing initial documents much. The idea is to focus on paraphrasing the existing information, making sure that extractive keyphrases are still recognizable. This approach may enhance the model's ability to identify repeated keyphrases while minimizing attention on extraneous content, which may be variable in nature.

One way to do this is to use a round-trip translation approach. The assumption under this strategy is that after translating some piece of information to various languages in a pipeline, the overall meaning will stay the same, but paraphrased [37].

The implementation strategy is mostly the same, as for new datasets generation, but instead of pre-trained backwards model only existing translation models should be used. The easiest approach is to translate the English text to some other language and then back. This can be repeated N times to get N new paraphrased copies of an original Inspec dataset. For experimentation, three foreign languages would be enough to get three copies of the original Inspec dataset. The chosen languages are: German, French and Spanish. For translation, the set of models from Helsinki-NLP seems to be the most convenient. However, they are based on the OPUS dataset [38], which is not specifically scientific as the Inspec dataset is and covers a wide amount of topics. This can be the problem if translating some poorly spreaded scientific terms, but there is no way to find any specifically scientific translation model.

So, the following models [39] were picked to create the paraphrased datasets:

1. Helsinki-NLP/opus-mt-en-de for English-German translation.
   Helsinki-NLP/opus-mt-de-en for German-English back-translation.
2. Helsinki-NLP/opus-mt-en-fr for English-French translation.
   Helsinki-NLP/opus-mt-fr-en for French-English back-translation.
3. Helsinki-NLP/opus-mt-en-es for English-Spanish translation.
   Helsinki-NLP/opus-mt-es-en for Spanish-English back-translation.

"document":
["A", "conflict", "between", "language", "and", "atomistic", "information", "Fred", "Dretske", "and", "Jerry", "Fodor", "are", "responsible", "for", "popularizing", "three", "well-known", "theses", "in", "contemporary", "philosophy", "of", "mind", ":", "the", "thesis", "of", "Information-Based", "Semantics", "-LRB-", "IBS", "-RRB-", ",", "the", "thesis", "of", "Content", "Atomism", "-LRB-", "Atomism", "-RRB-", "and", "the", "thesis", "of", "the", "Language", "of", "Thought", "-LRB-", "LOT", "-RRB-", ".", "LOT", "concerns", "the", "semantically", "relevant", "structure", "of", "representations", "involved", "in", "cognitive", "states", "such", "as", "beliefs", "and", "desires", ".", "It", "maintains", "that", "all", "such", "representations", "must", "have", "syntactic", "structures", "mirroring", "the", "structure", "of", "their", "contents", ".", "IBS", "is", "a", "thesis", "about", "the", "nature", "of", "the", "relations", "that", "connect", "cognitive", "representations", "and", "their", "parts", "to", "their", "contents", "-LRB-", "semantic", "relations", "-RRB-", ".", "It", "holds", "that", "these", "relations", "supervene", "solely", "on", "relations", "of", "the", "kind", "that", "support", "information", "content", ",", "perhaps", "with", "some", "help", "from", "logical", "principles", "of", "combination", ".", "Atomism", "is", "a", "thesis", "about", "the", "nature", "of", "the", "content", "of", "simple", "symbols", ".", "It", "holds", "that", "each", "substantive", "simple", "symbol", "possesses", "its", "content", "independently", "of", "all", "other", "symbols", "in", "the", "representational", "system", ".", "I", "argue", "that", "Dretske", "'s", "and", "Fodor", "'s", "theories", "are", "false", "and", "that", "their", "falsehood", "results", "from", "a", "conflict", "IBS", "and", "Atomism", ",", "on", "the", "one", "hand", ",", "and", "LOT", ",", "on", "the", "other"]

Figure 8. Example document from the default Inspec dataset (id 1015)

"document":
["A", "conflict", "between", "language", "and", "atomic", "information", "Fred", "Dretske", "and", "Jerry", "Fodor", "are", "responsible", "for", "the", "popularization", "of", "three", "well-known", "theses", "in", "the", "contemporary", "philosophy", "of", "mind", ":", "the", "thesis", "of", "Information-Based", "Semantics", "-LRB-", "IBS", "-RRB-", ",", "the", "thesis", "of", "Content", "Atomism", "-LRB-", "Atomism", "-RRB-", "and", "the", "thesis", "of", "Language", "of", "Thought", "-LRB-", "LOT", "-RRB-", ".", "LOT", "concerns", "the", "semantically", "relevant", "structure", "of", "representations", "in", "cognitive", "states", "such", "as", "convictions", "and", "desires", ".", "It", "claims", "that", "all", "these", "representations", "must", "reflect", "syntactical", "structures", "the", "structure", "of", "their", "contents", ".", "IBS", "is", "a", "thesis", "about", "the", "nature", "of", "relationships", "that", "connect", "cognitive", "representations", "and", "their", "parts", "with", "their", "contents", "-LRB-", "semantic", "relationships", "-RRB-", ".", "It", "notes", "that", "these", "relationships", "prevail", "exclusively", "on", "relationships", "of", "the", "kind", "that", "support", "information", "content,", "with", "the", "help", "of", "logical", "principles", "of", "combination.", ".", "Atomism", "is", "a", "thesis", "about", "the", "nature", "of", "the", "content", "of", "simple", "symbols", ".", "It", "holds", "that", "each", "essential", "simple", "symbol", "has", "its", "content", "independently", "of", "all", "other", "symbols", "in", "the", "representation", "system.", ".", "I", "argue", "that", "Dretske's", "and", "Fodor's", "theories", "are", "wrong", "and", "that", "their", "falsehood", "results", "from", "a", "conflict", "IBS", "and", "Atomism,", "on", "the", "one", "hand,", "and", "LOT,", "on", "the", "other", "hand"]

Figure 9. Example document after back-translation from German (id 1015)

"document":
["A", "conflict", "between", "language", "and", "atomistic", "information", "Fred", "Dretske", "and", "Jerry", "Fodor", "are", "responsible", "for", "popularizing", "three", "well-known", "theses", "in", "the", "contemporary", "philosophy", "of", "the", "mind:", "the", "thesis", "of", "information-based", "semantics", "-LRB-", "IBS", "-RRB-", ",", "the", "thesis", "of", "content", "atomism", "-LRB-", "Atomism", "-RRB-", "and", "the", "thesis", "of", "the", "language", "of", "thought", "-LRB-", "LOT", "-RRB-", ".", "LOT", "concerns", "the", "semantically", "relevant", "structure", "of", "the", "representations", "involved", "in", "cognitive", "states", "such", "as", "beliefs", "and", "desires", ".", "She", "argues", "that", "all", "these", "representations", "must", "have", "syntactic", "structures", "that", "reflect", "the", "structure", "of", "their", "content.", ".", "IBS", "is", "a", "thesis", "on", "the", "nature", "of", "the", "relationships", "that", "link", "cognitive", "representations", "and", "their", "parts", "to", "their", "content", "-LRB-", "semantic", "relations", "-RRB-", ".", "He", "believes", "that", "these", "relationships", "overlap", "only", "with", "relationships", "of", "the", "kind", "that", "support", "the", "content", "of", "the", "information,", "perhaps", "with", "some", "help", "of", "logical", "principles", "of", "combination.", ".", "Atomism", "is", "a", "thesis", "on", "the", "nature", "of", "the", "content", "of", "simple", "symbols", ".", "It", "considers", "that", "each", "simple", "background", "symbol", "has", "its", "contents", "independently", "of", "all", "other", "symbols", "of", "the", "representation", "system", ".", "I", "claim", "that", "the", "theories", "of", "Dretske", "and", "Fodor", "are", "false", "and", "that", "their", "lie", "is", "the", "result", "of", "a", "conflict", "between", "IBS", "and", "Atomism,", "on", "the", "one", "hand,", "and", "LOT,", "on", "the", "other", "hand."]

Figure 10. Example document after back-translation from French (id 1015)

"document":
["A", "conflict", "between", "language", "and", "atomistic", "information", "Fred", "Dretske", "and", "Jerry", "Fodor", "are", "responsible", "for", "popularizing", "three", "theses", "known", "in", "the", "contemporary", "philosophy", "of", "the", "mind:", "the", "thesis", "of", "Information-Based", "Semantics", "-LRB-IBS", "-RRB-,", "the", "thesis", "of", "Content", "Atomism", "-LRB-Atomism", "-RRB-", "and", "the", "the", "thesis", "of", "Thought", "Language", "-LRB-LOT", "-RRB-", ".", "LOT", "refers", "to", "the", "semantically", "relevant", "structure", "of", "representations", "involved", "in", "cognitive", "states", "such", "as", "beliefs", "and", "desires", ".", "He", "argues", "that", "all", "these", "representations", "must", "have", "syntactic", "structures", "that", "reflect", "the", "structure", "of", "their", "contents.", ".", "IBS", "is", "a", "thesis", "on", "the", "nature", "of", "relationships", "that", "connect", "cognitive", "representations", "and", "their", "parts", "with", "their", "contents", "-LRB-", "semantic", "relations", "-RRB-", ".", "He", "contends", "that", "these", "relationships", "are", "only", "supervening", "on", "relationships", "of", "the", "kind", "that", "support", "the", "content", "of", "the", "information,", "perhaps", "with", "some", "help", "from", "the", "logical", "principles", "of", "the", "combination.", ".", "Atomism", "is", "a", "thesis", "on", "the", "nature", "of", "the", "content", "of", "simple", "symbols", ".", "He", "argues", "that", "each", "single", "noun", "symbol", "has", "its", "content", "regardless", "of", "all", "other", "symbols", "in", "the", "representation", "system.", ".", "Argument", "that", "Dretske", "'s", "and", "Fodor", "'s", "theories", "are", "false", "and", "that", "their", "falsehood", "results", "from", "an", "IBS", "and", "Atomism", "conflict,", "on", "the", "one", "hand,", "and", "LOT,", "on", "the", "other"]

Figure 11. Example document after back-translation from Spanish (id 1015)

From the generated datasets it can be clearly seen that the resulting documents look nearly the same with minimum differences in phrasing (e.g. popularization/popularizing, false/wrong, lie/falsehood etc.). This difference results in the slight abstractive performance boost, described in the *Table 13*:

	Inspec (extractive)			**Inspec (abstractive)**		
	F@5	F@10	F@O	F@5	F@10	F@O
Shuffled sentences (previous best)	**45.5**	**53.3**	51.6	**19.6**	22.2	22.0
Translated dataset only	42.1	47.5	47.2	17.3	18.7	18.8
Translated dataset + 6x default Inspec	45.1	53.2	**52.9**	19.0	**22.5**	**22.8**

Table 13. Comparison of using the back-translated dataset with the previous best

This experiment evidently shows that the back-translation approach allows the model to perform much better than generating completely new samples, but the abstractive improvement is only a bit better than for the regular shuffled sentences approach, described in Chapter 5.2. The conclusion is that there is no need in changing initial training data semantic content, but the diversity of paraphrasing should be increased.

5.6 Experiment 6 - Synonyms processing

The only way to change the document phrasing significantly without changing semantics is to replace as many words as possible with their synonyms. The only criteria to avoid replacing the word is if it is used as a part of an extractive keyphrase to prevent the model from extracting a false-positive keyphrase.

The most convenient tool for this purpose is the NLTK (Natural Language Toolkit) library for Python with a WordNet extension package (an electronic lexical database [40]). This extensive library is designed to provide easy natural language processing, and is especially useful for the purpose of synonyms replacement.

To replace only related words with synonyms, the input document can be tokenized using the NLTK method word_tokenize to get tags for each token with method pos_tags. This method provides the detailed information about each token - grammatical category of the token (noun, verb, adjective, or adverb) and, optionally, the wordnet sense (unique identifier for the tokens' meaning). It is also possible to skip tokenization, as documents in the dataset are already splitted to separate words.

```
('Fred', 'NNP'), ('Dretske', 'NNP'), ('and', 'CC'), ('Jerry', 'NNP'), ('Fodor', 'NNP'), ('are', 'VBP'),
('responsible', 'JJ'), ('for', 'IN'), ('popularizing', 'VBG'), ('three', 'CD'), ('well-known', 'JJ'), ('theses',
'NNS'), ('in', 'IN'), ('contemporary', 'JJ'), ('philosophy', 'NN'), ('of', 'IN'), ('mind', 'NN')
```

Figure 12. Resulting tuples with grammatical categories of each token

Out of these POS tags, the only information needed to decide whether this word can be replaced by a synonym is the first character, which means part of the speech. For the current experiment, I'm going to replace only nouns (N), verbs (V), adjectives (J) and adverbs (R). Another check is to verify the word is not one of the extractive tokens, defined before using the list of current extractive keyphrases.

After the word is identified as suitable to be replaced, the wordnet library capabilities can be used to get a list of synonyms first, then rank them with a similarity score. This allows you to get N copies of the same text, which use different synonyms based on the similarity rank and the current allowable similarity index (the first copy should use only most relevant synonyms, then the second relevant synonym etc).

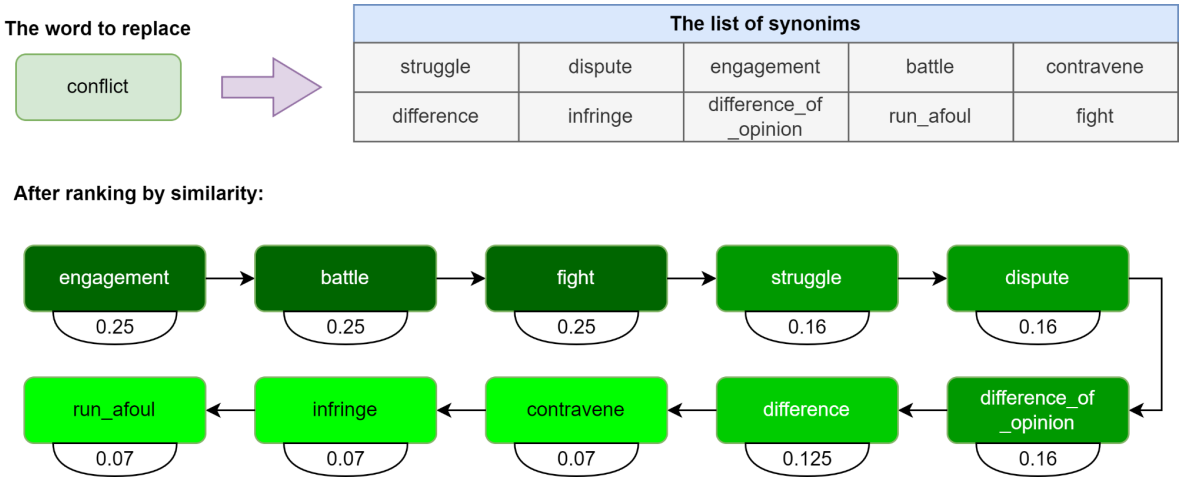

Figure 13. Example of ranking process for word synonyms

The result is that it becomes possible to get a large number of rephrased documents with the same structure and semantics:

Original: A conflict between language and atomistic information Fred Dretske and Jerry Fodor are responsible for popularizing three well-known theses in contemporary philosophy of mind.

- *Paraphrased (1st similarity rank):* A battle between language and atomistical info Fred Dretske and Boche Fodor constitute creditworthy for popularize three long-familiar dissertation in coeval philosophy of mind.
- *Paraphrased (2nd similarity rank):* A engagement between language and atomistical data Fred Dretske and Kraut Fodor exist responsible for vulgarise three long-familiar dissertation in present-day philosophy of mind.
- *Paraphrased (3rd similarity rank):* A fight between language and atomistical selective information Fred Dretske and Krauthead Fodor make up responsible for for popularise three long-familiar dissertation in modern-day philosophy of mind.
- *Paraphrased (4th similarity rank):* A struggle between language and atomistical entropy Fred Dretske and Hun Fodor live responsible for for vulgarize three long-familiar dissertation in contemporaneous philosophy of mind.
- *Paraphrased (5th similarity rank):* A difference of opinion between language and atomistical entropy Fred Dretske and Hun Fodor comprise responsible for generalise three long-familiar dissertation in contemporaneous philosophy of mind.

Figure 14. An example of paraphrased sentence from the Inspec document (id 1015)

Overall, in most cases even with the fifth rank of similarity the resulting paraphrased sentence looks meaningful and its context is still clear. The small mistakes like a wrong article in *"a engagement"* are allowable in this case, as they should not influence the process of keyphrase extraction and it would be much harder to avoid them using just an algorithmic processing with WordNet.

As a result the dataset with 5 paraphrased Inspec datasets (5000 samples) is ready and can be used for training after combination with either default dataset or any previously generated ones. The experiments with dataset combinations (original, generated, back-translated and synonyms) were not successful, as using the generated dataset always led to the noticeable decrease in performance, while using back-translated dataset was leading to overfitting and performance stagnation. At the end the best solution was to use only the original and

synonyms dataset, which led to a first significant improvement since using advanced shuffling techniques.

	Inspec (extractive)			Inspec (abstractive)		
	F@5	F@10	F@O	F@5	F@10	F@O
Shuffled sentences (previous best)	45.5	53.3	51.6	19.6	22.2	22.0
Synonyms dataset only (1 epoch)	39.6	47.7	47.5	17.6	20.5	20.4
Synonyms dataset + 3x default	**45.89**	**54.03**	**54.27**	**20.15**	**23.07**	**23.15**
Synonyms dataset + 3x default + shuffled sentences	43.9	52.3	51.9	19.3	22.3	22.5

Table 14. Comparison of using the synonyms dataset with the previous best

So, the best performance model acquired after training for 3 epochs on a combined synonyms and original dataset, and in comparison with the basic approach before applying any data augmentation technique the improvement is noticeable (*Table 15, Acquired compared to basic* row).

	Inspec (extractive)			Inspec (abstractive)		
	F@5	F@10	F@O	F@5	F@10	F@O
Basic approach	44.3	48.5	48.7	13.1	13.7	13.5
Basic approach (24 epochs)	**46.2**	52.1	52.2	5.6	15.5	15.3
Synonyms replacement	45.89	**54.03**	**54.27**	**20.15**	**23.07**	**23.15**
Acquired compared to basic	**+3.5%**	**+11.5%**	**+11.4%**	+53.8%	**+68.4%**	**+71.5%**
Acquired compared to basic (24 epochs)	-0.7%	+3.7%	+3.9%	**+359.8%** *(the reason explained further)*	+48.8%	+51.3%

Table 15. Relative performance boost, gained by applying advanced data augmentation techniques

Important to keep in mind that applying data augmentation is not the only difference with the basic approach, because often data augmentation leads to increasing the number of samples. For the basic approach the default dataset (Inspec) and epoch number (3) were used, while for the most advanced approach tested the combination of synonyms (5000 samples) and three default Inspec datasets (3000) were used with the default number of epochs. This way, the total number of samples for the basic approach is 3000, while for the most advanced one it reaches the relatively huge number - (5000 + 3000) * 3 = 24000. The same number of samples for basic approach is the equivalent for running training for 24 epochs, therefore there could be a concern that with such size of the training samples with constant dataset could give the same results, but here comes the overfitting problem.

For the purity of the experiment there is a point in running the basic approach without applying any data augmentation techniques and verify, whether the F1 score is reaching any reasonable performance rate. The results are listed above in the *Table 15, Acquired compared to basic (24 epochs)* row. *The extremely high relative improvement rate for abstractive F@5 is connected with the evaluation logic, which takes into account only top 5 predictions, which are more likely to be extractive keyphrases here. The reason for this behavior is that using a*

basic approach, keyphrases are just concatenated to a target training sequence as extractive_keyphrases + abstractive_keyphrases, which leads the model to be overfitted to predict extractive ones first on this number of epochs.

So, the experiment evidently showed that despite using a basic approach on the same amount of samples to achieve decent performance for extractive performance, it is still relatively poor for an abstractive one. So, when top 5 predictions are taken, there is a higher probability they are extractive and less abstractive ones are likely to fit there. The overfitting is also a reason for a poor overall abstractive performance, as the model starts paying too much attention to the explicitly mentioned keyphrases over a static context.

6 Discussion and future work

As a result, the best applied data augmentation technique was multiplying training data by replacing words with N-similar synonyms. Variable N in this case defines the number of dataset multiplications possible, but is limited by the number of suggested WordNet keyphrases. As the implementation considers similarity-ranked keyphrases array as circle-closed, the N value can be much bigger, leading to various combinations of synonyms in the new text. However, for the current work the number 5 was considered to be the best, as with this value it is almost guaranteed that the non-extractive noun, verb, adjective or adverb replaced will be unique for each newly generated dataset (as the number of synonyms suggested is usually bigger). The improvement after using the technique described was also proved by training a new model over the unmodified training data for the equivalent amount of samples. In this case extractive performance was not much worse, unlike the abstractive one, where using the dataset, modified with data augmentation technique proposed, led to a 51% relative improvement over the 24 epochs basic approach and 71.5% over the same approach with 3 epochs of training.

The underlying mechanism for the approach chosen to work most likely is due to the increasing the ability of the model to get the text context better by making relations between synonyms used stronger. Therefore, the model can extract the abstractive context from a broader specter of paraphrased sentences with the same meaning. On the other hand, the approach proposed also slightly increases the ability to spot extractive keyphrases, which probably can be explained by preserving them untouched in the newly generated texts, making them more outlined for the model among paraphrased surroundings. The actual numbers of relative improvement are 3% compared to the 24 epochs basic approach and 11.5% compared to the same approach with 3 epochs of training.

In future the system can still be improved by developing the idea of augmenting training data with synonyms by increasing the described N value (leading to a significant increase of newly generated training data, but resulting in less uniqueness of every separate text) or by applying more advanced paraphrasing techniques.

One of the possible approaches is to use some external model to get the ability to not only replace words with synonyms, but also apply a broader paraphrasing, keeping semantics the same with even larger possible number of word combinations. The main challenge I see for now with this proposed approach is whether it is necessary to aim at keeping extractive

keyphrases non-paraphrased or does ignoring this problem completely lead the model to even better extractive performance. In the case preserving extractive extractive keyphrases is considered to be unnecessary, the evaluation function should be modified to the initial implementation with special tokens in the target sequence (‘EXT:’ and ‘ABS:’), as the current implementation relies on extractive keyphrases being explicitly mentioned in the text to distinguish them from abstractive ones in the output sequence.

Another possible experiment would be to try any of the mentioned approaches on some language other than English. An interesting outcome could be whether mentioned methods provide the same performance boost to those languages and whether data augmentation techniques are universal for any language in the world.

7 Conclusion

In this work a lot of data augmentation techniques were presented and tested including the one, which gives significant performance boost for keyphrases detection. Replacing words with synonyms (*Chapter 5.6*) in the training data introduced an evident boost to the abstractive keyphrase extraction (up to 71.5% relative F1@O improvement, compared to the base approach, which can be seen in the *Table 15*), giving the model a broader ability to dig into the abstract context of the document by considering a large number of synonyms for various document phrasing. The resulting model was able to outperform simpler training methods and combined the machine learning and algorithmic NLP algorithms using NLTK and wordnet to achieve the result and generate a new dataset. It also worked quite well for extractive performance, leading to the relative improvement of 11.5% for F1@10. This means that the described synonyms approach can be used to get the significant abstractive performance boost without having to sacrifice the extractive one, but even increasing it a bit as well.

However, there were certain improvements spotted for other approaches as well. The MT-based approach also was able to give a slight relative abstractive performance boost, described in *Table 13*. The abstractive F1@O score improvement jumped from 22.0 to 22.8. So, the relative improvement was only 3.6%, however this was enough to show that introducing diversity of phrasings really makes a difference and this approach should be developed more to increase the rate of paraphrasing, as it was relatively small for the MT-based approach, described in the *Chapter 5.5*.

The easiest working data augmentation technique applied was a regular shuffling of keyphrases before every new epoch, described in the *Chapter 5.2*. The improvement, provided by using this technique (see *Table 8* for extractive and *Table 9* for abstractive) was the best for abstractive performance (F1@10 score 13.7 -> 23.7 for Inspec dataset and 4.9 -> 10.0 for Kp20k, the performance rate almost doubled), however extractive performance was also boosted (F1@10 score 48.5 -> 50.8 (+4.7%) for Inspec dataset and 24.4 -> 31.8 (+30.3%) for Kp20k). The reason for this approach to work so effectively for Kp20k is due to the size and unbalanced structure, leading the model to be overfitted to spot certain sequences with an unshuffled approach.

References

[1] Eberendu, Adanma. (2016). Unstructured Data: an overview of the data of Big Data. International Journal of Computer Trends and Technology. 38. 46-50. 10.14445/22312803/IJCTT-V38P109.

[2] Ghazi, Ahmed & Haggag, Mohamed. (2014). Keyword Extraction using Clustering and Semantic Analysis. International Journal of Science and Research (IJSR). 3. 1128-1132.

[3] M. Ramakrishna Murty, J. V. R. Murthy, P. V. G. D. Prasada Reddy & Suresh Chandra Satapathy (2012). Statistical Approach Based Keyword Extraction Aid Dimensionality Reduction. Springer Berlin Heidelberg. 10.1007/978-3-642-27443-5_51

[4] Qaiser, Shahzad & Ali, Ramsha. (2018). Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. International Journal of Computer Applications. 181. 10.5120/ijca2018917395.

[5] François Rousseau and Michalis Vazirgiannis. 2013. Graph-of-word and TW-IDF: new approach to ad hoc IR. Proceedings of the 22nd ACM international conference on Information & Knowledge Management (CIKM '13). Association for Computing Machinery, New York, NY, USA, 59–68. https://doi.org/10.1145/2505515.2505671

[6] Decui Liang, Bochun Yi, Wen Cao, Qiang Zheng, Exploring ensemble oversampling method for imbalanced keyword extraction learning in policy text based on three-way decisions and SMOTE, Expert Systems with Applications, Volume 188, 2022, 116051, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2021.116051.

[7] Shibata, Tomohide & Kato, Norio & Kurohashi, Sadao. (2007). Automatic object model acquisition and object recognition by integrating linguistic and visual information. 383-392. 10.1145/1291233.1291327.

[8] Der, Lonneke & Pallotta, Vincenzo & Rajman, Martin & Ghorbel, Hatem. (2004). Automatic Keyword Extraction from Spoken Text. A Comparison of two Lexical Resources: the EDR and WordNet.

[9] Glazkova, Anna & Morozov, Dmitry. (2022). Applying Transformer-based Text Summarization for Keyphrase Generation. 10.48550/ARXIV.2209.03791

[10] Yousuf, Hana & Gaid, Michael & Salloum, Said & Shaalan, Khaled. (2020). A Systematic Review on Sequence to Sequence Neural Network and its Models.

International Journal of Electrical and Computer Engineering. 11. 10.11591/ijece.v11i3.pp2315-2326.

[11] Markus Freitag and Yaser Al-Onaizan. (2017). Beam Search Strategies for Neural Machine Translation. 10.18653/v1/w17-3207

[12] Chowdhury, Md. Faisal Mahbub & Rossiello, Gaetano & Glass, Michael & Mihindukulasooriya, Nandana & Gliozzo, Alfio. (2022). Applying a Generic Sequence-to-Sequence Model for Simple and Effective Keyphrase Generation.

[13] Kulkarni, Mayank and Mahata, Debanjan and Arora, Ravneet and Bhowmik, Rajarshi. (2021). Learning Rich Representation of Keyphrases from Text, arXiv:2112.08547

[14] H. Dong, J. Wan and Z. Wan, "Keyphrase Extraction Based on Multi-Feature," 2019 International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI), Taiyuan, China, 2019, pp. 208-213, doi: 10.1109/MLBDBI48998.2019.00047.

[15] N. T. A. Meem, M. M. H. Chowdhury and M. M. Rahman, "Keyphrase Extraction from Bengali Document using LSTM Recurrent Neural Network," 2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT), Dhaka, Bangladesh, 2018, pp. 461-466, doi: 10.1109/CEEICT.2018.8628090.

[16] E. Doostmohammadi, M. H. Bokaei and H. Sameti, "Persian Keyphrase Generation Using Sequence-to-Sequence Models," 2019 27th Iranian Conference on Electrical Engineering (ICEE), Yazd, Iran, 2019, pp. 2010-2015, doi: 10.1109/IranianCEE.2019.8786505.

[17] E. Doostmohammadi, M. H. Bokaei and H. Sameti, "PerKey: A Persian News Corpus for Keyphrase Extraction and Generation," 2018 9th International Symposium on Telecommunications (IST), Tehran, Iran, 2018, pp. 460-465, doi: 10.1109/ISTEL.2018.8661095.

[18] 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.

[19] Sparapani R, Spanbauer C, McCulloch R (2021). “Nonparametric Machine Learning and Efficient Computation with Bayesian Additive Regression Trees: The BART R Package.” Journal of Statistical Software, 97(1), 1–66. doi:10.18637/jss.v097.i01.

[20] L. Zhang and Y. Hu, "A fine-tuning approach research of pre-trained model with two stage," 2021 IEEE International Conference on Power Electronics, Computer Applications (ICPECA), Shenyang, China, 2021, pp. 905-908, doi: 10.1109/ICPECA51329.2021.9362566.

[21] Meng, Rui and Zhao, Sanqiang and Han, Shuguang and He, Daqing and Brusilovsky, Peter and Chi, Yu. (2017). Deep Keyphrase Generation. In proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 582-592, Vancouver, Canada. Association for Computational Linguistics, http://aclweb.org/anthology/P17-1054

[22] Lee Xiong and Chuan Hu and Chenyan Xiong and Daniel Fernando Campos and Arnold Overwijk. (2019). Open Domain Web Keyphrase Extraction Beyond Language Modeling. In EMNLP.

[23] Gallina, Ygor and Boudin, Florian and Daille, Beatrice. (2019). KPTimes: A Large-Scale Dataset for Keyphrase Generation on News Documents. In Proceedings of the 12th International Conference on Natural Language Generation, pages 130-135

[24] Mikalai Krapivin and Aliaksandr Autaeu and Maurizio Marchese. (2009). Large Dataset for Keyphrases Extraction.

[25] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov and Luke Zettlemoyer. (2019). BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. http://arxiv.org/abs/1910.13461

[26] A. Zheng and A. Casari, Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists. Sebastopol, CA, USA: O’Reilly Media, Inc., 2018.

[27] B. Wang and N. Z. Gong, "Stealing Hyperparameters in Machine Learning," 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2018, pp. 36-52, doi: 10.1109/SP.2018.00038.

[28] K. Kim, "Time-Sensitive Adaptation of Regularization Strength of Recurrent Neural Networks for Accurate Learning," 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 2017, pp. 194-198, doi: 10.1109/ICMLA.2017.00036.

[29] J. Han and J. Kim, "Selective Data Augmentation for Improving the Performance of Offline Reinforcement Learning," 2022 22nd International Conference on Control, Automation and Systems (ICCAS), Jeju, Korea, Republic of, 2022, pp. 222-226, doi: 10.23919/ICCAS55662.2022.10003747.

[30] P. Benz, C. Zhang and I. S. Kweon, "Batch Normalization Increases Adversarial Vulnerability and Decreases Adversarial Transferability: A Non-Robust Feature Perspective," 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 2021, pp. 7798-7807, doi: 10.1109/ICCV48922.2021.00772.

[31] K. Taneja and J. Vashishtha, "Comparison of Transfer Learning and Traditional Machine Learning Approach for Text Classification," 2022 9th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2022, pp. 195-200, doi: 10.23919/INDIACom54597.2022.9763279.

[32] Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 582–592, Vancouver, Canada. Association for Computational Linguistics

[33] Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Peter Brusilovsky, Daqing He, and Adam Trischler. 2020. One size does not fit all: Generating and evaluating variable number of keyphrases. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7961– 7975, Online. Association for Computational Linguistics

[34] Qianying Liu, Wenyu Guan, Sujian Li, Fei Cheng, Daisuke Kawahara and Sadao Kurohashi. (2021). Reverse Operation based Data Augmentation for Solving Math Word Problems. https://arxiv.org/abs/2010.01556.

[35] ML6Team. (2021). Keyphrase Extraction model using distilbert as a base and fine-tuned on the Inspec dataset. [Token Classification]. Hugging Face's model hub. Referenced on April 30, 2023.
https://huggingface.co/ml6team/keyphrase-extraction-distilbert-inspec

[36] Victor Sanh, Lysandre Debut, Julien Chaumond and Thomas Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter", 2019, arXiv:abs/1910.01108.

[37] Jean-Philippe Corbeil and Hadi Abdi Ghadivel. (2020). BET: A Backtranslation Approach for Easy Data Augmentation in Transformer-based Paraphrase Identification Context. https://arxiv.org/abs/2009.12452

[38] Biao Zhang, Philip Williams, Ivan Titov and Rico Sennrich. (2020). Improving Massively Multilingual Neural Machine Translation and Zero-Shot Translation. https://arxiv.org/abs/2004.11867

[39] [OPUS-MT – Building open translation services for the World] (https://aclanthology.org/2020.eamt-1.61) (Tiedemann & Thottingal, EAMT 2020)

[40] Fellbaum C (1998). WordNet: An Electronic Lexical Database. Bradford Books. https://mitpress.mit.edu/9780262561167/

[41] Bharti, Drsantosh & Babu, Korra. (2017). Automatic Keyword Extraction for Text Summarization: A Survey. http://arxiv.org/abs/1704.03242.

[42] A. Dridi, H. Afifi, H. Moungla and C. Boucetta, "Transfer Learning for Classification and Prediction of Time Series for Next Generation Networks," ICC 2021 - IEEE International Conference on Communications, Montreal, QC, Canada, 2021, pp. 1-6, doi: 10.1109/ICC42927.2021.9500507.

Appendix 1 - Links to the codebase and the best model along with the modified dataset

Codebase for experiments can be found at:
https://github.com/lordofelectrons/keyphrase-extraction-scripts

Data augmented Inspec dataset for synonyms approach can be found at:
https://huggingface.co/datasets/artemfilipenko/synonyms-augmented-5x-inspec

The best model, trained on a combined default and data augmented Inspec dataset is publicly accessible here:
https://huggingface.co/artemfilipenko/keyphrase-generation-bart-large-trained-on-augmented-and-default-inspec

Appendix 2 - Non-exclusive license for reproduction and publication of a graduation thesis[1]

I, Artem Filipenko

1. Grant Tallinn University of Technology free license (non-exclusive license) for my thesis “Data Augmentation Techniques for Advanced End to End Keyphrase Extraction From Text”, supervised by Tanel Alumäe.
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive license.
3. I confirm that granting the non-exclusive license does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

07.05.2023

[1] The non-exclusive license is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive license, the non-exclusive license shall not be valid for the period.