This document is part of the Research and Innovation Action "European Live Translator (ELITR)".
This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 825460.

Deliverable D5.1

Initial Report on Summarization

Erion Çano, Anja Nedoluzhko, Ondřej Bojar (CUNI)

Dissemination Level: Public

Final (Version 1.0), 30th June, 2020

Grant agreement no.	825460
Project acronym	ELITR
Project full title	European Live Translator
Type of action	Research and Innovation Action
Coordinator	Doc. RNDr. Ondřej Bojar, PhD. (CUNI)
Start date, duration	1st January, 2019, 36 months
Dissemination level	Public
Contractual date of delivery	30th June, 2020
Actual date of delivery	30th June, 2020
Deliverable number	D5.1
Deliverable title	Initial Report on Summarization
Type	Report
Status and version	Final (Version 1.0)
Number of pages	47
Contributing partners	CUNI
WP leader	CUNI
Author(s)	Erion Çano, Anja Nedoluzhko, Ondřej Bojar (CUNI)
EC project officer	Alexandru Ceausu
The partners in ELITR are:	▪ Univerzita Karlova (CUNI), Czech Republic ▪ University of Edinburgh (UEDIN), United Kingdom ▪ Karlsruher Institut für Technologie (KIT), Germany ▪ PerVoice SPA (PV), Italy ▪ alfatraining Bildungszentrum GmbH (AV), Germany
Partially-participating party	▪ Nejvyšší kontrolní úřad (SAO), Czech Republic

For copies of reports, updates on project activities and other ELITR-related information, contact:

Doc. RNDr. Ondřej Bojar, PhD., ÚFAL MFF UK
Malostranské náměstí 25
118 00 Praha, Czech Republic
bojar@ufal.mff.cuni.cz
Phone: +420 951 554 276
Fax: +420 257 223 293

Copies of reports and other material can also be accessed via the project's homepage:

`http://www.elitr.eu/`

Contents

1 Executive Summary

This deliverable reports on the progress in the minuting activities related to WP5. In this work package, we will design and implement a method for automatic summarization of speech transcript into the form of structured meeting minutes, populating a pre-defined structured agenda. The process is conceived as a pipeline of subtasks which are Task 5.1: Meeting Segmentation (automatic separation and clustering of various types of utterances, CUNI; months 1–12), Task 5.2: Segment-Level Summarization (summarizing sentences into bullet points, UEDIN, CUNI, KIT; months 7–24), Task 5.3: Document-level Summarization (selecting segments to include in the summary, CUNI; months 13–24), and Task 5.4: Sequence to Structure (matching the bullet points with the meeting agenda topics, CUNI; months 19–36).

Each subtask will be evaluated intrinsically. The overall integration and testing will happen in WP6. As shown in Table 1, Task 5.1 was concluded according the the plan and the other tasks are on track.

Task	**Months**	**Status**
5.1: Meeting Segmentation	1–12	CONCLUDED AS PLANNED
5.2: Segment-Level Summarization	7–24	AS PLANNED
5.3: Document-level Summarization	13–24	AS PLANNED
5.4: Sequence to Structure	19–36	AS PLANNED

Table 1: Execution Timeline of the Minuting Tasks

The followind sections provide details on each of the tasks. Additionally, we report on the progress of the construction of a corpus of meeting recordings in Section 3, despite it formally belongs to WP1.

2 The Minuting Pipeline

One of the research goals on ELITR is to come up with methods of "minuting", i.e. automatic summarization of meetings. This goal is a natural continuation of other tasks explored in ELITR, e.g. support for multi-lingual discussion.

The minuting aims to automate the extraction of the important pieces of information (e.g., the decisions or conclusions) from the meeting transcript and structures them based on the meeting agenda topics. The minuting module takes as input the meeting transcript and the "Agenda" (a list of topics to be discussed) and it is expected to produce the minutes of the meeting and the filled agenda (a topic match between some of the minutes and the agenda points). It is conceived as a pipeline of four steps as depicted in Figure 1. Our code, data and preliminary results are available in the online repository.[1] The following subsections describe each step in more details.

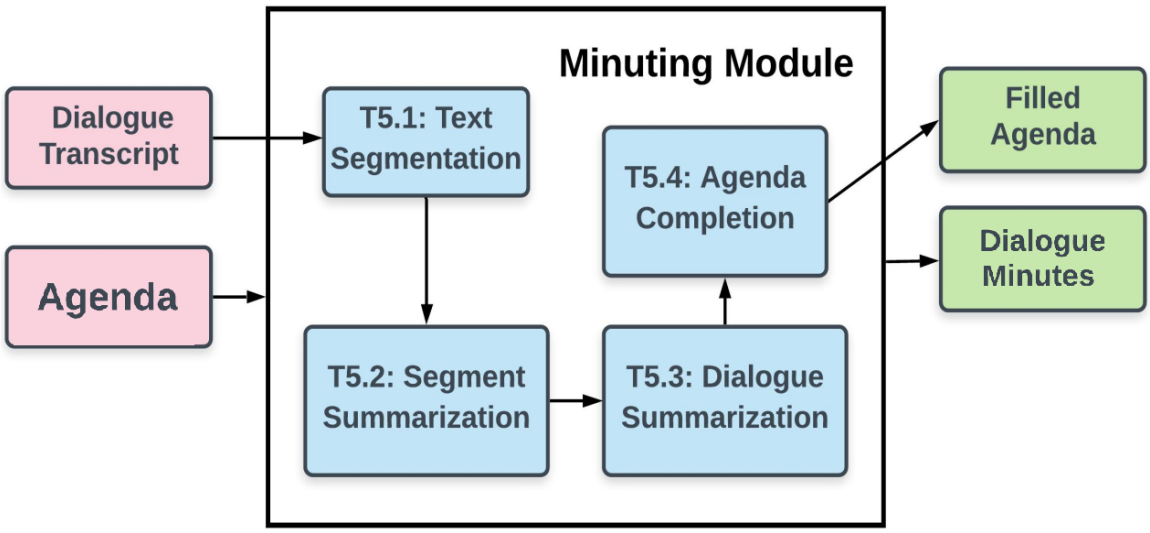

Figure 1: Schematic view of the minuting module.

2.1 Preliminary Research Activities

The main part of the minuting module is the topic aggregation of the transcript segments and the topic matching with the agenda clauses. For this reason, we started the minuting research activity trying to understand how are the topic aspects of a document reflected in its main metadata like *title*, *abstract* and *keywords*. Çano and Bojar (2019a) brings a survey on the keyword extraction and generation. This work is reproduced in Appendix A and presents several insights about the recent techniques involved in the process, the most popular keyword generation datasets, the strategies that are used for keyword quality evaluation, etc.

Another issue we wanted to explore was the relation between the summarization task and the document keywords. We checked the quality of keywords produced by the abstractive summarization of document *titles* and *abstracts* (Çano and Bojar, 2019c). The most important take away message we got from this work (reproduced in Appendix B) is the fact that abstractive summarization techniques do not properly preserve the topic aspects of their text imputs. For this reason, we are inclined to use extractive techniques in the minuting process, despite trying abstractive ones as well.

In a final empirical survey, we explored various text summarization techniques from the data efficiency point of view (Çano and Bojar, 2019b) and came to realize that the Transformer architecture (Vaswani et al., 2017) is the most data efficient from the ones we tried. Furthermore, this work (reproduced in Appendix C) brings various text summarization ROUGE scores

[1] `https://github.com/ELITR/minuting-experimentation`

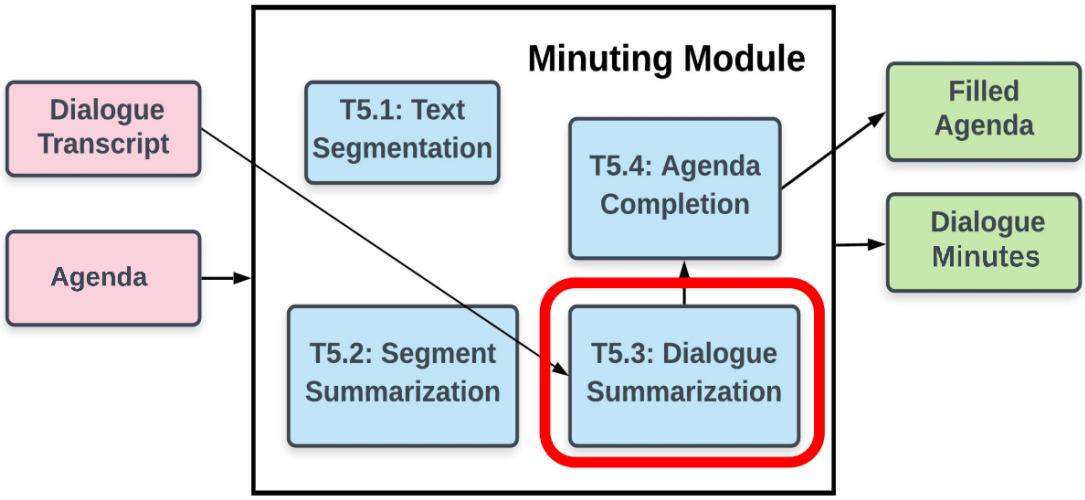

Figure 2: Attempt to shortcut the minuting process.

(Lin, 2004) that have served us as comparison baselines on our ongoing text summarization experiments.

2.2 Data Inputs and Outputs

The inputs of the minuting module are the meeting transcript and the predefined agenda with its topics which were planned to be discussed by the participants. The dialogues in the transcript are probably messy since they come out of the Automatic Speech Recognition module. There will likely be language disfluencies that are typical in spoken language.

The predefined agenda is just a list topics that are to be discussed in the transcript. It may happen that certain topics of the agenda are not discussed at all. Other unpredicted topics (not listed in the agenda) may also be discussed and be present in the transcript.

The best publicly available datasets of transcripts and minutes that we found are the AMI[2] and ICSI[3] corpora. AMI is a collection of 134 transcripts and 100 hours of audio recordings, together with various valuable annotations. ICSI is smaller, containing 58 transcripts and 70 hours of recordings. They were both released in 2005 and have become very popular in the minuting research literature. Additionally, we are building our own corpus of meetingss, see Section 3. The problem with these datasets is their small size. Since we are trying big models based on neural networks, we obviously needed larger text collections. To overcome this limitation, we crawled several academic networks and created two corpora (OAGKX and OAGSX) of publication metadata that are described by Çano and Bojar (2020), reproduced in Appendix D. We also adopted the CNN/DailyMail dataset of Nallapati et al. (2016) that is very popular in the literature.

Another preliminaty data-related activity we performed was the transformation of AMI and ICSI corpora from their XML format to a plaintext format that is much easier to work with. This process also removed many of their details such as emotional or body language annotations that are not used in the minuting process.

Finally, before running any experiment, we performed a few more text cleaning steps. We lowercased everything to reduce the reduce the vocabulary size and used Standord CoreNLP (Manning et al., 2014) to tokenize the dialogues in words.

2.3 T5.1: Transcript Segmentation (months 1–12)

Before organizing the minuting task in subtasks, one thing we wanted to try was to make sure that it was really necessary. In other words, we wanted to be sure that trying to obtain the

[2] http://groups.inf.ed.ac.uk/ami/download/
[3] http://groups.inf.ed.ac.uk/ami/icsi/download/

transcript minutes directly from the transcript (as it is shown in Figure 2) with existing text summarization models would not produce acceptable results. We applied a transfer learning approach the the Transformer of Vaswani et al. (2017) as the text summarizer, the CNN/Dailymail news stories as train samples and the AMI data as test samples. As we expected, the results were very poor both quantitatively (very low ROUGE scores) and qualitatively (messy and incorrect minutes).

To improve the quality of the transcript minutes, we conceived the minuting process as the pipeline illustrated in Figure 1. The next step after the data preprocessing is the segmentation of the whole meeting transcript. The goal of this step is to break the spoken language texts into segments which are a couple of consecutive sentences normally related to the same topic. This step is delicate since we have to take care of the logical and grammatical correctness of the produced segments (e.g., correctly identifying sentence boundaries).

First, we tried a supervised approach of Koshorek et al. (2018). The task is formulated as binary classification of sentences; namely, each sentence is assigned a probability that it is the last one in the current segment. The dataset consists of about 700k Wikipedia articles, where the segments are given by the top-level hierarchy of each document. A single data example is a document represented as a list of sentences. The model consists of two levels of LSTM cells. The first level operates independently on each sentence in the document to create a sentence representation. The second level then operates on these sentence representations and equips them with some contextual information. We replicated the results (Koshorek et al., 2018) and marginally improved them by modifying the training objective to reflect the Pk metric that is used to evaluate performance of segmentation models. However, the main disadvantage of this approach is the assumption of a large in-domain dataset, which is, in our circumstances, problematic. Therefore, we also investigated other methods.

Since there is no limitation in the number of the segments that are created, we use the intra-segment semantic similarity to optimize that number. The transcript is first split in separate sentences. Then we group them in various clusters and check the average intra-cluster cosine similarity value. The final segment configuration (the number of segments and the phrases in each of them) is the one that maximizes this value. Until now we have explored several unsupervised learning algorithms. According to our results, agglomerative clustering works best, at least for the data we are working with.

Another possible subtask is the diarization which identifies the speaker's name in the text. We can have this name as a label in front of each text segment. For the moment, we haven't conducted any diarization experiments. Meanwhile, the AMI and ICSI corpora are already diarized with dummy names such as letters like A, B, C, D. So one possibility is that we perform a dummy diarization just to maintain compatibility with AMI and ICSI data. In the long term, we plan to apply a real diarization.

2.4 T5.2: Segment Compression (months 7–24)

The next step is segment summarization, or summarization applied in each of the discourse segments. This is technically very similar to sentence compression, even though we have several sentences in each cluster. Our goal here is to further improve the text quality by removing the disfluencies and redundancies of the spontaneous speech. We also want to preserve the original information and to ensure the grammatical correctness of the test segments. This is achieved by applying extractive (to preserve original phrasing) models based on LSTM neural networks that delete redundant words by pruning the syntactic tree of the entire sentence. These supervised learning approaches such as Filippova et al. (2015) or Lai et al. (2018) are very popular in the literature. There are still several recent attempts with unsupervised methods that rely on contextual word embeddings such as those of pretrained language models (Zhou and Rush, 2019). They achieve compelling results without the need for any paird data samples. As for the datasets, most of the studies use big corpora like the one of Filippova and Altun (2013) or the annotated English Gigaword of Napoles et al. (2012).

2.5 T5.3: Transcript Summarization (months 13–24)

The next step is the transcript summarization subtask which produces one of the desired outputs, the meeting minutes. The goal in this phase is to correctly identify the most important phrases from the segments. They can be sentences that express meeting decisions or take-away messages. Once again, grammatical correctness and logical order of the phrases in the minute is mandatory. We are exploring both extractive and abstractive approaches for this task. Since we need to archive the meeting decisions, the extractive way seems more convenient. The biggest difficulty we are facing is the lack of domain data. AMI and ICSI are very small to train big models such as the Transformer. We are actually using them as test sets only. For the training we are using the news texts of the CNN/Dailymail collection.

The recent solutions for this task are based on pretrained models such as BERT and offer the possibility to create both extractive and abstractive models (Liu and Lapata, 2019). We used the extractive model trained on CNN/Dailymail data reached the authors' scores[4] (ROUGE-1: 42.33, ROUGE-2: 19.54, and ROUGE-L: 38.78 F1 scores) when testing on the test split of the same dataset. However, that was not the case when testing on the minuting samples of AMI and ISCI corpora. We reached ROUGE-1: 16.24, ROUGE-2: 4.33, and ROUGE-L: 14.07 F1 scores which are considerably lower. From the qualitative point of view, we observed that the summaries contain a few word repetitions and many grammatical errors. The most probable explanation for these poor results is the domain difference between the news of CNN/Dailymail and the meeting transcripts of AMI and ICSI. Overcoming this handicap is our biggest challenge for the moment.

2.6 T5.4: Agenda Completion (months 19–36)

The last step of the pipeline is the completion of the predefined agenda. It will produce the second output that is the agenda with clauses from the minutes matched to its topics. The matching of the minute phrases and the agenda topics will be achieved by applying topic modeling methods and maximizing their respective semantic similarities. The progress on this pipeline phase depends on the quality of the discourse summaries we obtain from the previous phases and the accuracy of the topic modeling solutions.

We are also working on the minuting demonstrator and its design. This is a tool conceived as an autonomous script that reads the growing transcript and uses the minuting module to populate the agenda with the minutes. Every time that the transcript grows with a certain amount of words (e.g., the script could be activated for every 400 words added to the transcript), it summarizes those words and populates the agenda. Then it waits for the next sequence of words. Ideally, the demonstrator and its interface would work live during the meetings.

[4] `https://github.com/nlpyang/PreSumm`

3 Corpus of Minuting, Work in Progress

The data creation is primarily the part of the WP1 Data but because there is no delierable from WP1, we are reporting on the progress of the Minuting corpus creation here in D5.1.

Within the ELITR project, we create the corpus of minuting. The corpus consists of meetings in English and Czech and the minutes to these meetings. For the meetings, we have their audio or video recordings, automatic ASR transcripts and manually corrected transcripts. Some meetings are equipped with double-checked manual transcripts. As for the minutes, we have original minutes, completed by the meeting organizer or a secretary, and then we provide the additionally created minutes by our annotators. For some meetings, we create multiple minutes, to study the mechanism of important information extraction and summarization.

The annotation of the minuting corpus is going on. For the time being, we have the numbers which are presented in Table 2. The numbers in the table are not cumulative, i.e., for example, for English meetings, we have the general collection consisting of 87 hours of the recordings, out of which 40 hours have been provided with manual transcripts and 37 hours contain also the specially generated minuted.

Meetings	**English**	**Czech**
meetings with original minutes	87h. (70 meetings)	46h. (42 meetings)
meetings with manual transcripts	40h. (42 meetings)	41h. (37 meetings)
meetings with additional annotation of minutes	37h. (39 meetings)	41h. (37 meetings)

Table 2: Corpus of Minuting

At the moment, we don't have the deidentification of the meetings yet, so the texts may contain some personal information.

The meetings are also different as concerns the consents of the participants with their use in the future. Most of the meetings will be open, but some will be used only internally.

4 Conclusion

This deliverable described the progress in WP5 Minuting and also the related data acquisition (WP1) for meeting summarization.

Overall, WP5 proceeds according to the plans but given the complexity of the meeting summarization task and the insatisfactory results of the baselines examined so far, we do not expect that we can arrive at a fully working solution by the end of the project. Instead, anticipate WP5 to conclude with precisely defining the task and providing a solid dataset for testing and limited training. We plan to organize a shared task to allow also external teams to contribute to the development of meeting summarization methods.

References

Erion Çano and Ondřej Bojar. Keyphrase generation: A multi-aspect survey. In *2019 25th Conference of Open Innovations Association (FRUCT)*, pages 85–94, Helsinki, Finland, Nov 2019a. doi: 10.23919/FRUCT48121.2019.8981519. URL `https://ieeexplore.ieee.org/document/8981519`.

Erion Çano and Ondřej Bojar. Efficiency metrics for data-driven models: A text summarization case study. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 229–239, Tokyo, Japan, October–November 2019b. Association for Computational Linguistics. doi: 10.18653/v1/W19-8630. URL `https://www.aclweb.org/anthology/W19-8630`.

Erion Çano and Ondřej Bojar. Keyphrase generation: A text summarization struggle. 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 666–672, Minneapolis, Minnesota, June 2019c. Association for Computational Linguistics. doi: 10.18653/v1/N19-1070. URL `https://www.aclweb.org/anthology/N19-1070`.

Erion Çano and Ondřej Bojar. Two huge title and keyword generation corpora of research articles. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 6663–6671, Marseille, France, may 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://www.aclweb.org/anthology/2020.lrec-1.823.

Katja Filippova and Yasemin Altun. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1481–1491, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1155.

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Lukasz Kaiser, and Oriol Vinyals. Sentence compression by deletion with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 360–368, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1042. URL https://www.aclweb.org/anthology/D15-1042.

Omri Koshorek, Adir Cohen, Noam Mor, Michael Rotman, and Jonathan Berant. Text segmentation as a supervised learning task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 469–473, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2075. URL https://www.aclweb.org/anthology/N18-2075.

Dac-Viet Lai, Nguyen Truong Son, and Nguyen Le Minh. Deletion-based sentence compression using bi-enc-dec lstm. In Kôiti Hasida and Win Pa Pa, editors, *Computational Linguistics*, pages 249–260, Singapore, 2018. Springer Singapore. ISBN 978-981-10-8438-6.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Proc. ACL workshop on Text Summarization Branches Out*, page 10, 2004. URL http://aclweb.org/anthology/W04-1013.

Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1387. URL https://www.aclweb.org/anthology/D19-1387.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-5010. URL https://www.aclweb.org/anthology/P14-5010.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290. Association for Computational Linguistics, 2016. doi: 10.18653/v1/K16-1028. URL http://aclweb.org/anthology/K16-1028.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-Scale Knowledge Extraction*, AKBC-WEKEX '12, page 95–100, USA, 2012. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Jiawei Zhou and Alexander Rush. Simple unsupervised summarization by contextual matching. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5101–5106, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1503. URL https://www.aclweb.org/anthology/P19-1503.

A Keyphrase Generation: A Multi-Aspect Survey

PROCEEDING OF THE 25TH CONFERENCE OF FRUCT ASSOCIATION

Keyphrase Generation: A Multi-Aspect Survey

Erion Çano, Ondřej Bojar
Charles University
Prague, Czech Republic
{cano, bojar}@ufal.mff.cuni.cz

***Abstract*—Extractive keyphrase generation research has been around since the nineties, but the more advanced abstractive approach based on the encoder-decoder framework and sequence-to-sequence learning has been explored only recently. In fact, more than a dozen of abstractive methods have been proposed in the last three years, producing meaningful keyphrases and achieving state-of-the-art scores. In this survey, we examine various aspects of the extractive keyphrase generation methods and focus mostly on the more recent abstractive methods that are based on neural networks. We pay particular attention to the mechanisms that have driven the perfection of the later. A huge collection of scientific article metadata and the corresponding keyphrases is created and released for the research community. We also present various keyphrase generation and text summarization research patterns and trends of the last two decades.**

I. Introduction

A keyphrase or a keyword (here we use them interchangeably) is a short set of one or a few words that represent a concept or a topic covered in a document. They are commonly used to annotate articles or other documents and are essential for the categorization and fast retrieval of such items in digital libraries. A keyphrase string, on the other hand, is a set of comma-separated (other separators may be used as well) keyphrases associated with an article or a different type of object, describing the content and topical aspects of it.

Because of their high importance and the need to process huge amounts of documents with missing keyphrases, KG (Keyphrase Generation) attracted high academic interest since the 90s. Some basic works of that time such as [1], [2], and [3] used text features and supervised learning algorithms (popular at that time) to extract keywords from documents. Improved supervised methods like [4], [5], and [6], graph-based methods like [7], [8], and [9], or other unsupervised KG methods such as [10] and [11] were proposed in the 2000s.

Extractive KG became so popular in the 2000s and early 2010s, that the entire research field was commonly called KE (Keyphrase Extraction). This success was mainly due to the simplicity and speed of the proposed solutions. There is still a serious flaw in extractive KG: the inability to produce absent keyphrases (predicted keyphrases that do not appear in the source text). Analyzing the most popular datasets, [12] showed that present (predicted keyphrases that also appear in the source text) and absent keyphrases assigned by paper authors are almost equally frequent. Ignoring the later is thus a serious handicap.

Motivated by the advances in sequence-to-sequence applications of neural networks, several studies like [12] or [13] started to explore AKG (Abstractive Keyphrase Generation). The encoder-decoder (or sequence-to-sequence learning) paradigm that was first utilized in the context of machine translation (e.g., in [14], [15] or [16]) got quick adaption in related tasks such as text summarization (like in [17] and [18]) or AKG. Since that time, AKG research took over and is today a vibrant field of study.

In this survey, we start by reviewing the most popular KE methods, specifically the supervised, the graph-based and the other unsupervised ones. We go on describing the popular existing keyphrase datasets and present OAGKX, a novel and huge collection of about 23 million metadata samples (titles, abstracts, and keyphrase strings) from scientific articles that is released online (http://hdl.handle.net/11234/1-3062). It can be used as a data source to train deep supervised KG methods or to create byproducts (other keyphrase datasets) from more specific scientific disciplines.

Unlike similar recent reviews such as [19], [20], or [21] that focus entirely on extractive KG (or KE), the main interest of this work in the more recent and technically advanced AKG studies which are examined in details. Particular attention is paid to the network structures and the enhancement mechanisms, as well as to the evaluation process the authors follow. We also describe certain research patterns that we observed such as the interesting analogy with similar developments in text summarization research.

II. Extractive Keyphrase Generation

Extractive keyphrase generation methods are simpler and appeared in the literature in the late 90s. They usually follow two steps. First, candidate phrases are selected from the document. Different strategies are latter applied to decide if each candidate is a keyphrase or not. The following subsections briefly describe the most popular extractive methods. More comprehensive and detailed reviews entirely focused on KE can be found in other surveys such as [19].

A. Supervised Methods

One of the first studies that considered KG as a supervised learning problem was [3]. In that study, the author experimented with texts from journal articles, email messages, and Web pages. Some of the used features were *word frequency*, *phrase length*, *number of words in phrase*, etc. C4.5 decision tree algorithm of [22] was utilized as a classifier, in combination with a bagging procedure based on random sampling with replacement presented at [23]. The author also experimented with GenEx, an algorithm described in [2] that was specifically designed to extract document keyphrases. He concluded that domain knowledge is highly valuable in the keyphrase extraction process and GenEx (using that knowledge) performs significantly better than C4.5 (not using it).

ISSN 2305-7254

This work encouraged other researchers to develop supervised learning methods for solving KE problems.

Almost at the same time, KEA (Keyphrase Extraction Algorithm), a language-independent supervised KG algorithm was developed and presented in [1]. It uses features like *TF-IDF* and *first occurrence* and then applies Naïve Bayes classifier to determine if candidate phrases are keyphrases or not. Authors evaluate KEA using NZDL dataset (see Table I) and report that it is able to correctly identify one or two of the first five author keyphrases.

The development of Maui, a similar algorithm presented in [5] was a further step forward. Maui extends KEA in several ways. It combines more feature types and exploits Wikipedia articles as a source of linguistic knowledge. Furthermore, Maui can work well with both Naïve Bayes and bagged decision tree classifiers.

Some attempts explore various feature setups for improving the existing methods. The author in [24] investigates the role of additional features like n-grams, *noun phrases*, *POS tags*, etc. She concludes that using words or n-grams that match *POS tag* patterns increases the recall compared to the usage of n-grams only. Furthermore, according to her results, the syntactic information of the POS tags is also important for optimizing the number of keyphrases assigned to each document. The author also creates and uses Inspec, a dataset of scientific paper metadata. Other studies like [25] or [26] that followed also experimented with scientific paper texts, a practice that is common even today.

The logical structure of a scientific article is defined in [27] as the hierarchy of its logical components like title, list of authors, abstract and sections. Authors of [6] use that logical structure to build WINGNUS by limiting the number of identified candidate phrases. They further use different features like *length of phrases*, *typeface* and *position* (in title, introduction, etc.) for training the Weka implementation of Naïve Bayes (presented in [28]) to select the best candidates. Authors conclude that using the logical structure of the scientific articles yields superior performance over methods that do not consider that information.

In [29] they experimented by adding syntactic relations extracted with the dependency parser of [30]. They also tried different classifiers like Support Vector Machines of [31] and Random Forests of [32]. According to their results, the NLP-based features improve F_1 scores of all the tested methods. They also concluded that Random Forest is a good trade-off between keyphrase quality and generation speed.

There were also a few studies that applied neural network structures to perform extractive KG. In [4], for example, they used a feed-forward neural network as a classifier and paid particular attention to title headings (also subheadings) and phrase repetitions. Authors of [33], on the other hand, utilized a more complex neural network structure based on LSTMs (Long Short-Term Memory) to build an end-to-end keyphrase extraction system that eliminates the need for manual engineering of statistical features.

B. Graph-Based Methods

From the unsupervised extractive KG methods, those based on graph computations are the most numerous. In [34] they introduced TextRank, a graph-based ranking method inspired by the PageRank algorithm of [35]. They implement the idea of "voting": a vertex that represents a word or phrase (lexical unit) links to another one, casting a vote in the later. A higher number of votes to certain words or phrases suggests that they are more important. All lexical units of the source text are ranked this way. The returned keyphrases are constructed from the top N words.

Authors of [7] use the concept of the neighborhood of a given document: a set of similar documents that expands that document. They later employ PageRank on the local graph (of a single document) or the expanded graph (of the neighborhood) to rank the words and phrases. SingleRank and ExpandRank are the names of the corresponding methods they derive. The authors report that ExpandRank is significantly better than SingleRank for any size of the neighborhood.

In [36] they follow a similar approach to formulate CiteTextRank. Authors use the documents citing the given document (citation network) to expand it and then they apply PageRank. TopicRank defined in [9] is another improvement over TextRank. It first clusters lexical units of the document according to their topic. Afterwards, it uses a graph-based ranking model to assign scores to the topic clusters. Finally, keyphrases are generated by picking one of them from each ranked cluster.

One of the fastest available KE methods is RAKE proposed in [8]. Authors first remove punctuation and stop words and then create a graph of word co-occurrences. Candidate words are scored based on the degree and frequency of each word vertice in the graph. The top-scoring ones are returned as keyphrases. Authors report that RAKE achieves higher precision and similar recall when compared with other graph-based methods like TextRank.

PositionRank is yet another graph-based KE approach recently proposed by [37]. They construct a word-level graph where they incorporate information from positions of all word occurrences. PageRank is later used to score the words and phrases. Authors show that using positions of all word occurrences works better than using the first occurrence of each word only.

C. Other Methods

Besides the two categories above, there are also other unsupervised methods that are not graph-based. They mostly utilize clustering and various similarity measures to find the best keyphrases. A very simple scheme uses TF-IDF to compute scores and rank text phrases of the entire document. This raw approach is one of the most frequent baselines in other studies that propose KG methods.

Authors of [38] proposes another basic approach based on term frequencies and stopword filtering. In [10] they argue that KG systems should be unsupervised and domain-independent. They build a KG system based on loosely structured ontologies. Authors of [39] rely on Deep Belief Networks described

in [40] to capture the intrinsic representations of documents and using them to extract keyphrases.

Another peculiar approach is the one by [41] who consider keyphrasing as a form of translation from the language of the document to the language of keyphrases. They use word alignment from statistical machine translation to learn matching probabilities between document words and keyphrase words.

Statistical language models are also used by [42] who utilize Kullback-Leibler divergence described in [43] to create a single score (including phraseness and informativeness) for ranking extracted phrases. YAKE! presented in [11] is another example of an unsupervised and feature-based extractive KG solution. They utilize features like *casing*, *word position*, *word frequency*, and more, combined in a complex scoring function that is used to yield the ranked keyphrases.

There is also a recent attempt in [44] to use the concept of word embeddings in the context of the unsupervised KE. Authors propose Key2Vec, a method for training phrase (multi-word) embeddings which are used to represent the candidate keyphrases and build the thematic representation of the document. The candidate keyphrases are later ranked based on their thematic relation with the document using the theme-weighted PageRank algorithm of [45].

The many extractive (supervised, graph-based or other) KG methods described in this section are complementary and may be used in different scenarios and for different purposes. To ease their implementation and benchmarking, the author of [46] created PKE, a Python toolkit available online (https://github.com/boudinfl/pke). It implements many of the above methods, offers pretrained and ready to use KE models and can also be easily extended to implement or benchmark new methods.

III. Keyphrase Datasets

A. Popular Corpora

The recent open data initiatives and data science competitions have encouraged the creation and sharing of more and more datasets. There are papers like [47] that release data about movies, [48] about music, [49] about books and [50] that describes data of other object categories. The computational linguistics or natural language processing datasets consist of various text collections that are used to solve particular tasks. In the realm of KG, the most popular in the literature are the collections of scientific articles shown in Table I.

Inspec is one of the earliest datasets, released in [24] where the role of various linguistic features in KE is explored. It consists of 2000 paper titles (1500 for training and 500 for testing), abstracts and keywords from journals of Information Technology, published from 1998 to 2002.

One of the smallest is NUS of [51], consisting of 211 conference papers. Each paper has two sets of keyphrases: one set by the authors and a second that was created by volunteer students. Another small dataset is SemEval (or SemEval-2010) described in [52]. It is composed of 244 papers, 144 for training and 100 for testing. They were collected from ACM Digital Library and belong to conference and workshop proceedings.

TABLE I. Public keyphrase datasets

Reference	Name	Content	# Docs
[24] Hulth	Inspec	Papers	2000
[51] Nguyen	NUS	Papers	211
[52] Kim	SemEval	Papers	244
[7] Wan	DUC	News	308
[29] Krapivin	Krapivin	Papers	2304
[53] Zhang	Twitter	Tweets	147K
[12] Meng	KP20k	Papers	567K
[1] Witten	NZDL	Reports	1800

Krapivin, the dataset released in [29] has the advantage of providing full paper texts together with the corresponding metadata. There is a total of 2304 Computer Science articles published by ACM from 2003 to 2005. The parts of each text such as title, abstract and sections are separated and marked to ease the extraction of various keyphrases.

The most popular KG dataset of the recent years is probably KP20k released in [12]. It consists of 567830 Computer Science articles, 527830 for training, 20K for validation and 20K for testing. KP20k has been used for training and evaluating various recent abstractive methods. The biggest keyphrase dataset is probably OAGK recently released in [59]. It contains 2.2M titles, abstracts and keyphrase strings of scientific papers from different disciplines.

The above scientific paper datasets are summarized in Table I. There are also a few more datasets of other document types, but they are less popular in the literature. One of them is NZDL, a collection of 1800 Computer Science technical reports, 1300 for training and 500 for testing. It is described in [1]. Authors use it to benchmark KEA, their extractive method which was one of the first.

From the news domain, the DUC (or DUC-2001) dataset of [7] is somehow popular. It consists of 308 news articles and 2048 keyphrase labels and has been used in a few extractive and abstractive KG methods. In [53] they create a dataset of about 147K tweets and their corresponding tags. Authors use it to evaluate their model for hashtag prediction. Authors of [54] use a dataset of 815 Web pages and the corresponding extracted keywords for addressing advertisements.

The two most recent datasets are probably StackExchange (post topics) and TextWorld (game observations and commands) created and used by [55]. Similar datasets can be found in other works like [56], [57] or [58].

B. A Novel and Huge Data Collection

Experimenting with keyphrases of scientific papers seems an ongoing trend that is greatly motivated by the availability of data in online academic repositories. Following the examples of [59] and [60], we took the initiative to produce an even larger collection of scientific paper keywords, titles and abstracts. Exploiting the whole data of Open Academic Graph (described in [61] and [62]), we retrieved *keywords*, *title* and *abstract* data wherever they were available. A language filter was applied to remove every text record not in English. We also lowercased and utilized Stanford CoreNLP of [63] to tokenize the *title* and *abstract* texts.

TABLE II. TOKEN STATISTICS OF OAGKX

Attribute	Title	Abstract	Keywords
Total	290 M	4 B	270 M
Min / Max	3 / 25	50 / 400	2 / 60
Mean	12.8 (4.9)	175.1 (86.5)	11.9 (7.5)
Overlaps	78 % (17 %)	68 % (25 %)	

TABLE III. KEYWORD STATISTICS OF OAGKX

Attribute	Value
Total	133 295 056
Min / Max	2 / 12
Mean	5.9 (3.1)
Present	52.7 % (28.3 %)
Absent	47.3 % (28.3 %)

Since there were several articles with very short or very long text fields (outliers), we removed any record with a title not within 3-25 tokens, abstract not within 50-400 tokens or keyphrase strings not within 2-60 tokens. We also removed records with a number of keyphrases now within 2-12. The obtained dataset is OAGKX, a collection of about 23 million article metadata records.

Some basic statistics regarding the distribution of tokens in *title*, *Abstract* and *Keywords* fields of the articles can be found in Table II. As we can see, the average lengths are about 13 tokens for the titles, 175 tokens for the abstracts, and 12 tokens for the keyphrase strings (standard deviation is given in parenthesis). We also computed the token overlaps between abstracts and titles, and between abstracts and keyphrase strings. The overlap $o(s,t) = \frac{|\{s\} \cap \{t\}|}{|\{t\}|}$ between two token vectors s (source) and t (target) is the fraction of unique tokens in t that overlap with a source token in s. As we can see, there is high repetition of abstract words, both in titles (78 %) and in keyphrases (68 %).

We further observed the distribution of keyphrases. The corresponding statistics are shown in Table III. There is a total of about 133 million keyphrases with an average of about 6 in each article. The minimal and maximal of keyphrases in each record is 2 and 12 respectively. In KG experiments, it is also important to check the frequencies of the keyphrases that are present and absent in the source texts. The present fraction $p(s,k) = \frac{|k \cap s|}{|k|}$ is the fraction of the keyphrases k that do appear in the source text s. The absent fraction $a(s,k) = \frac{|k| - |k \cap s|}{|k|}$ is the its complement, or in other words the fraction of the keyphrases k that do not appear in the source text s. As we can see, OAGKX present and absent keyphrases are almost equally frequent (52.7 % vs. 47.3 %). This is in line with the observation of [12].

Using three extractive methods described in Section II, We performed some preliminary experiments with OAGKX data. We picked YAKE!, RAKE and TopicRank which are simple and used them with their default parameters in each implementation. Given that they are unsupervised and require test data only, we picked a big test cut of 100K samples from the entire OAGKX. In addition to the preprocessing steps described above which were performed on entire OAGKX collection, we also replaced digit symbols with # and joined each title and abstract in common source string. The length of this source string was limited to 260 tokens (a paper abstract and the title should not be longer).

For the evaluation, we used F_1 score of full matches between predicted keyphrases from each method and those available in the data record (author keyphrases). We computed F_1 scores on top 5, top 7 and top 10 returned keywords. Before comparing, both sets of terms were stemmed with Porter Stemmer and duplicates were removed. The obtained results are presented in Table IV. As we can see, the best of the three methods is YAKE, with top $F_1@10$ score of 21.86 %. We also observed that RAKE was considerably faster than the two other methods.

To have an idea about the topic distribution of OAGKX articles, we inspected a few randomly picked data records. We noticed that they belong to various scientific disciplines, with medicine (and its research directions) being dominant. There are also many papers about economics, social sciences or different technical disciplines. To our best knowledge, this is the biggest available collection of scientific paper data and the corresponding keyphrases. The value of OAGKX is thus twofold: (i) It can supplement the existing datasets if more training data are required. (ii) It can serve as a data source for creating scientific article subsets of more specific scientific disciplines or domains.

IV. ABSTRACTIVE KEYPHRASE GENERATION

In this section, the recent AKG methods based on the encoder-decoder framework are examined in detail. Table V summarizes some of their neural network properties, together with the evaluation data and metrics used by the authors.

A. Basic Neural Network Models

The authors of [53] were among the first to try RNNs (Recurrent Network Networks) for generating keyphrases (actually hashtags) of tweets. They adopt a joint-layer RNN with two hidden layers and two output ones. The latter are combined to form the objective layer (keyword or not). Authors build and refine a big dataset of tweets and the corresponding hashtags (keywords in this context) for evaluating their method. The basic LSTM of [64] and AKET, a tool for keyword extraction on tweets described in [65] are used as comparison baselines. Superior scores of 80.74 %, 81.19 % and 80.97 % are reported in terms of P (Precision), R (Recall) and F_1 respectively.

Another important work is [12], the first to adapt the encoder-decoder framework for AKG. Their CopyRNN model has an encoder that creates a hidden representation of the source text and a decoder that generates the keyphrases based on that representation. They employ a bidirectional GRU of [14] as the encoder and a forward GRU as the decoder. Keyphrase generation involves a beam search described in [66] with max depth 6 and beam size 200. The attention mechanism of [66] and copying mechanism of [67] are implemented to improve performance and alleviate the out-of-vocabulary words problem.

Authors evaluate CopyRNN on Inspec, Krapivin, NUS and SemEval and KP20k (IKNSK for short) datasets. Comparing with previous extractive approaches, they report state-of-the-art results in terms of $F_1@5$ (0.328 on KP20k) and $F_1@10$

TABLE IV. KE SCORES ON OAGKX (100K)

Method	F_1@5	F_1@7	F_1@10
YAKE!	19.27	21.49	21.86
RAKE	14.39	17.51	18.22
TopicRank	16.68	20.12	20.14

(0.255 on KP20k) scores for present keyphrases. They also report top scores on R@10 and R@50 for absent keyphrases. Their work created a roadmap of using the encoder-decoder framework for AKG that has been followed by many other researchers in these last three years.

In [68] they tried to optimize the speed of CopyRNN building CopyCNN made up of CNNs (Convolutional Neural Networks) which work in parallel. CNN layers are stacked on top of each other to process variable-length input text representations and gated linear units are used as the non-linearity function, same as in [69]. They also use position embeddings combined with input word embeddings to preserve the sequence order. Authors test their method using IKNSK and compare against several extractive methods and CopyRNN. They report slightly higher performance scores (in F_1@5, F_1@10, R@10, and R@50) compared to CopyRNN of [12]. Their model is also considerably faster, with generation times at least 6.2x lower.

Furthermore, authors in [13] tried to improve another aspect of CopyRNN, handling of keyword repetitions during generation. They build their model (CovRNN) utilizing a bidirectional GRU for encoding and a forward GRU for decoding. To consider the correlation of the generated target keyphrases with each other (avoiding repetitions), they implement the coverage mechanism of [70]. Same data (training on KP20k and evaluation on IKNSK) setups are used. The authors compare against extractive methods and CopyRNN. They report slightly better results compared to CopyRNN on both present (using F_1@4 and F_1@8) and absent (using R@10 and R@50) keyphrases.

B. Enhanced and Hybrid Solutions

Many works followed, improving different aspects of AKG. Authors of [71] propose a solution for handling repetition and increasing keyphrase diversity. Besides using coverage, they also implement a review mechanism that considers the source context as well as a target context (collection of hidden states) before predicting (decoding) the next keyphrase. Same as above, they implement their model (CorrRNN) with bidirectional GRU, forward GRU and beam search. They utilize the training part of KP20k and evaluate on NUS, SemEval and Krapivin datasets, comparing against several extractive methods and CopyRNN. Given that keyphrase diversity is important, besides the typical F_1 and R metrics, they also utilize α-NDCG of [72]. The authors report improvements on all reported metrics. Peak scores of 0.318 in F_1@5 and 0.278 in F_1@10 are reached on Krapivin dataset. They also assess the generalization ability of their model by training it with articles and testing it on news using DUC dataset.

All the above methods are supervised and depend on labeled training data which are not available for certain domains. In [73] they try to overcome this limitation using two approaches. In the first one, they tag unlabeled documents with synthetic keyphrases obtained from unsupervised methods and use them for model pretraining. The pretrained model is later tuned on the labeled data. In the second one, they use multitask learning by combining the task of AKG based on labeled data with the task of title generation (a form of text summarization) on unlabeled data.

Both tasks are implemented with a bilinear LSTM as the encoder and a plain LSTM as the decoder. In the multitask learning case, the encoder is shared by the two tasks wheres the decoders are different. Authors use KP20k as a source of labeled and unlabeled data and evaluate on IKNSK. A cross-domain test with news data (DUC dataset) is also performed. Their models outrun CopyRNN on all reported metrics (F_1@5, F_1@10 and R@10) reaching a peak score of 0.308 in F_1@5 on KP20k test set.

Authors of [74] try to inject the power of extraction and retrieval into the encoder-decoder framework. A neural sequence learning model is used to compute the probability of being a keyword for each word in the source text. Those values are later used to modify the copying probability distribution of the decoder, helping the later to detect the most important words. They also use a retriever to find documents annotated similarly which provide external knowledge for the decoder and guide the generation of the keyphrases for the given document. Finally, a merging module puts together the extracted, retrieved, and generated candidates, producing the final predictions. The authors use the same data and evaluation setup as above. They report superior scores of 0.317 in F_1@5 and 0.282 in F_1@10 for present keyphrases as well as significant improvements in R@10 scores for absent keyphrases.

Furthermore, in [75], they emphasize the important role of article title which indeed can be considered as a high-level summary of the text. Their solution (TG-Net) uses a complex encoder made up of three main parts. First, a bidirectional GRU is used to separately encode the source text (abstract + title) and the title in their corresponding contextual representations. Second, a matching layer catches the relevant title information for each context word using their semantic relation. Finally, another bidirectional GRU merges the original context and the gathered title information into a final title-guided representation. The decoder is similar to the ones described above, equipped with attention and copying. The authors train with KP20k and test on IKNSK. They report important gains over CopyRNN and CopyCNN on present keyphrases, with top scores 0.372 in F_1@5 and 0.315 in F_1@10 on KP20k test set. They also report significant improvements in absent keyphrases (higher R@10 and R@50 scores).

An attempt to improve KG diversity is found in [76] where their method produces keyphrases one at a time, considering the formerly generated keyphrases. This is achieved by using multiple decoders (each of them generates only one keyphrase) that focus on different words of the source text by subtracting the attention value derived from the previous decoder. As a result, beam searches of beam size 1 are used to get the top keyphrase from each decoder and coverage is used to have diverse words in each keyphrase. The authors train their model with KP20k (the train split) and test on Inspec, Krapivin, and KP20k (the test split). They report improvements on keyphrase diversity measured using distinct-1 and distinct-2 metrics described in [77].

TABLE V. SUMMARY OF AKG MODEL PROPERTIES. IKNSK = {INSPEC, KRAPIVIN, NUS, SEMEVAL-2010, KP20K}, NSK = {NUS, SEMEVAL-2010, KRAPIVIN}, IKK = {INSPEC, KRAPIVIN, KP20K}, GT = GENERATION TIME.

	Method				Evaluation	
Reference	**Network**	**Att**	**Copy**	**Cov**	**Data**	**Metrics**
[53] Zhang2016	joint-layer, RNN	-	-	-	Tweets	Precision, Recall, F_1
[12] Meng2017	Enc-Dec, GRU	✓	✓	-	IKNSK	$F_1@5$, $F_1@10$, R@10, R@50
[68] Zhang2017	Enc-Dec, CNN	✓	✓	-	IKNSK	$F_1@5$, $F_1@10$, R@10, R@50, GT
[13] Zhang2018	Enc-Dec, GRU	✓	✓	✓	IKNSK	$F_1@4$, $F_1@8$, R@10, R@50
[71] Chen2018a	Enc-Dec, GRU	✓	✓	✓	NSK	$F_1@5$, $F_1@10$, R@10, N@5, N@10
[73] Ye2018	Semisup, LSTM	✓	✓	-	IKNSK	$F_1@5$, $F_1@10$, R@10
[75] Chen2018b	Enc-Dec, GRU	✓	✓	-	IKNSK	$F_1@5$, $F_1@10$, R@10, R@50
[74] Chen2019	Hybrid, GRU	✓	✓	-	IKNSK	$F_1@5$, $F_1@10$, R@10
[76] Misawa2019	MultiDec, GRU	✓	✓	✓	IKK	$F_1@5$, $F_1@10$, dist1, dist2
[78] Wang2019	NTM, GRU	✓	✓	-	Blogs	$F_1@1$, $F_1@3$, $F_1@5$
[55] Yuan2018	catSeq, LSTM	✓	✓	-	IKNSK	$F_1@5$, $F_1@10$, $F_1@M$, $F_1@V$
[79] Chan2019	RL, GRU	✓	✓	-	IKNSK	$F_1@5$, $F_1@M$

In [78] they create another hybrid system that infuses topical information into the encoder-decoder framework. They use an NTM (Neural Topic Model) for grasping the latent topic aspects of the input text. The later go into the decoder, together with the context representation of the input obtained by the encoder. Their learning objective is modified accordingly to balance the effects of the NTM and the KG encoder-decoder. Authors conduct experiments on blog data such as Twitter, Weibo (a Chinese microblogging website) and StackExchange. They compare tag prediction of their method against various previous methods such as CopyRNN, TG-Neg, and CorrRNN, reporting considerable improvements in terms of $F_1@1$, $F_1@3$ and $F_1@5$ scores.

All the above works generate a fixed number of keyphrases per document. This is not optimal and realistic. In real scientific literature, different documents are paired with keyphrase sets of different lengths. To overcome this limitation and further improve the diversity of the produced keyphrases, authors of [55] propose a seq2seq generator equipped with advanced features. They first join a variable number of key terms as a single sequence and consider it as the target for sequence generation (sequence-to-concatenated-sequences or catSeq). By decoding a single of those sequences for each sample (e.g., taking top beam sequence from beam search) their model can produce variable-length keyphrase sequences for each input sample.

For a higher diversity in output sequences, they apply orthogonal regularization on the decoder hidden states, encouraging them to be distinct from each other. Authors use the same data setup as in [12] and compare against CopyRNN and TG-Net. Besides using $F_1@5$, $F_1@10$, they also propose two novel evaluation metrics: $F_1@M$, where M is the number of all keyphrases generated by the model for each data point, and $F_1@V$, where V is the number of predictions that gives the highest $F_1@V$ score in the validation set. Considerable improvements are achieved in terms of $F_1@10$ (top score 0.361), $F_1@M$ (top score 0.362) and $F_1@V$ (top score 0.362) on KP20k test set.

C. Reinforcement Learning Perspective

Given that the above catSeq model tends to generate fewer keywords than the ground-truth, authors of [79] reformulate it from the RL (Reinforcement Learning) perspective which has also been applied recently in several text summarization works like [80], [81] or [82] and similar seq2seq applications described in [82]. The model is stimulated to generate enough keyphrases employing an adaptive reward function that is based on recall (not penalized by incorrect predictions) in undergeneration scenarios and F_1 (penalized by more incorrect predictions) in overgeneration scenarios. They use GRU instead of LSTM but keep most of the other implementation details the same as those of [55].

The authors train on KP20k and test on IKNSK. They compare the RL-implemented catSeq, CopyRNN, and TG-Net against their original versions and report improvements from the RL implementation in all cases on both $F_1@5$ and $F_1@M$ with peak scores 0.321 and 0.386 respectively. The RL perspective is thus highly effective for enhancing existing AKG methods. Another contribution of their work is the novel comparison scheme they propose, with name variation sets for each ground-truth keyphrase. If a predicted keyphrase matches any name variation of a ground-truth keyphrase, it is considered as a correct prediction.

V. KEYPHRASING RESEARCH PATTERNS

There are several patterns regarding technical and other aspects of research that show up from time to time. In this section, we briefly summarize some of such trends we identified in KG and TS (Text Summarization) research of the last two decades.

A. Experimental Patterns

All of the primary studies we consulted perform some text preprocessing steps such as tokenization and lowercasing. Most papers do not report the tokenization utility they use. A few of them like [75] and [78] report to have used Stanford CoreNLP of [63] or NLTK (www.nltk.org) for tokenizing. It is also common to find KE studies like [46], [24], and [9] that perform POS tagging and include the tags in the feature set they utilize.

A reduced vocabulary size is important to have decent AKG resutls within a reasonable computation time. For this reason, authors of many recent AKG studies like [12], [71], [68], [73], [75] and [79] replace all digit tokens with the symbol ⟨digit⟩. Stemming is also commonly used in studies

like [12], [75], [74], [24], [71] and [73] to have the predicted and golden keywords properly compared during evaluation. A stemmer that is reported is the one of [83]. There are still a few works like [13] that do not report to use stemming or any other transformation in the evaluation step.

The motivation or objective of the authors is the same in most of the studies: producing meaningful and accurate keyphrases that are similar to those set by humans which are used as ground-truth. Besides that, there are a few studies such as [76] or [71] that aim for a higher diversity or avoiding duplicates in the produced keyphrases. Producing a different number of keyphrases for each document is another requirement. It was met just recently by the model of [55].

Overcoming the need for labeled or domain-specific data was also important for certain studies like [73] and [10]. Few works such as [8] and [68] focus on computational efficiency and generation speed while trying to keep state-of-the-art accuracy. Other works such as [79] and [33] are based on neural networks and attempt to generate more keyphrases (the former) or automate feature crafting (the latter). Finally, [46] creates a framework for implementing popular methods instead of proposing a new one.

All studies do perform a formal evaluation of their contribution with the exception of [11] where they highlight the functional features of their method by means of a practical demonstration. In the evaluation phase, they usually compare with similar methods used as baselines. Regarding the choice of baselines, we observed a similar trend in both extractive and abstractive KG studies. The earlier extractive works such as [1], [6] or [26] do not compare against other methods. In few cases such as in [24] and [7], they compare different versions (or configuration choices) of their basic method.

The more recent extractive works like [1], [8], [34], [29], [33], [36], and [37] compare against the earlier ones. Similarly, the earlier abstractive KG studies such as [12] and [53] compare against extractive methods only. Instead, some of the latest abstractive works such as [75], [13] or [79] compare against both extractive and abstractive KG methods.

B. Keyphrasing vs. Summarizing

Some interesting research patterns we observed are related to the strict analogy between the dynamics of TS and KG research in the last two decades. Extensive research began in the late 90s on both tasks. Early TS works were mostly extractive, same as the KG works of the same time (commonly called KE studies). They were usually based on lexical resources and features, clustering algorithms and similarity measures (e.g., [84], [85] or [86]). Several supervised TS works such as [87] and [88] or graph-based TS works like [89], [90] and [91] bloomed, in full analogy with the KG works of Sections II-A and II-B.

The same development path has been followed in the case of abstractive studies as well. The encoder-decoder framework equipped with attention was first used by [92] for title generation. In analogy with the studies of Section IV-B, many studies like [17] or [93] added copying mechanism whereas [18] was the first that used coverage. All these innovations significantly improved the results. The trend towards the RL approach makes no exception. It was first introduced in text summarization studies like [81] and [94]. As described in Section IV-C, It has been applied in AKG just recently.

There are still a few differences between TS and KG research that are related to the nature of these tasks. First, as presented in Section III-A, KG research works have mostly used scientific paper data. TS studies, on the other hand, have been mostly based on news articles (e.g., [95], [96] or [97]). In fact, most of the popular TS datasets like those described in [98], [99], and [93] are made up of online news articles preprocessed by the authors.

Another difference lies in the metrics that are used to perform the evaluation of the two tasks. KG methods are usually assessed by means of F_1 and recall whereas TS studies use more complex scores such as ROUGE of [100] or sometimes even BLEU of [101].

VI. Discussion

This study presents a survey of the earlier extractive KG methods and the recent cutting-edge abstractive ones that are based on the encoder-decoder framework. We first describe in brief some of the pivotal KE works which are supervised, unsupervised or graph-based. They were very successful and shaped the research field in the 2000s, mainly because of their speed and simplicity.

We then present the available keyphrase datasets that are popular in the literature and describe OAGKX, a huge article data collection that is released with this paper. It can be used as a data supplement for training deep learning models that require millions of samples. It might as well serve as a source for creating derivative datasets of scientific articles from more specific research disciplines.

The shift to the recent abstractive methods was mainly pushed from the need to annotate documents with keyphrases that do not necessarily appear in the original text. The availability of the easy-to-implement encoder-decoder framework was another motive. Advanced mechanisms such as attention, copying and coverage were added one by one and improved not only the accuracy but also the diversity of the produced keyphrases.

We further observed several similar patterns between TS and KG research. They include the transit from extractive to abstractive strategies, the use of technically advanced mechanisms (e.g., attention, copying, and coverage), and the reformulation of the methods from the reinforcement learning perspective. The latter trend is very promising and we expect to see many works in the near future exploring it in several ways for achieving different goals.

Acknowledgment

This research work was supported by the project No. CZ.02.2.69/0.0/0.0/16 027/0008495 (International Mobility of Researchers at Charles University) of the Operational Programme Research, Development and Education, the project no. 19-26934X (NEUREM3) of the Czech Science Foundation and ELITR (H2020-ICT-2018-2-825460) of the EU.

REFERENCES

[1] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, C. G. Nevill-Manning, "Kea: Practical automatic keyphrase extraction", *In Proceedings of the Fourth ACM Conference on Digital Libraries*, DL '99, pages 254–255, 1999.

[2] P. Turney, "Learning to extract keyphrases from text", unpublished.

[3] P. Turney, "Learning algorithms for keyphrase extraction", *Information Retrieval*, 2(4):303–336, May 2000.

[4] J. Wang, H. Peng, J. Hu, "Automatic keyphrases extraction from document using neural network", *Advances in Machine Learning and Cybernetics*, pages 633–641, 2006.

[5] O. Medelyan, *Human-competitive automatic topic indexing*. The University of Waikato, Phd Thesis, 2009.

[6] T. D. Nguyen, M. Luong, "Wingnus: Keyphrase extraction utilizing document logical structure", *In Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 166–169, Stroudsburg, PA, USA, 2010.

[7] X. Wan, J. Xiao, "Single document keyphrase extraction using neighborhood knowledge", *In Proceedings of the 23rd National Conference on Artificial Intelligence*, AAAI '08, Volume 2, pages 855–860, 2008.

[8] S. Rose, D. Engel, N. Cramer, W. Cowley, "Automatic keyword extraction from individual documents", *Text Mining. Applications and Theory*, pages 1–20, 2010.

[9] A. Bougouin, F. Boudin, B. Daille, "Topicrank: Graph-based topic ranking for keyphrase extraction", *In Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 543–551, 2013.

[10] D. D. Nart, C. Tasso, "A domain independent double layered approach to keyphrase generation", *In Proceedings of International Conference on Web Information Systems and Technologies*, 2014.

[11] R. Campos, V. Mangaravite, A. Pasquali, A. M. Jorge, C. Nunes, A. Jatowt, Brin, "Yake! collection-independent automatic keyword extractor", *Advances in Information Retrieval*, pages 806–810, Springer International Publishing, 2018.

[12] R. Meng, S. Zhao, S. Han, D. He, P. Brusilovsky, Y. Chi, "Deep keyphrase generation", *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 582–592, 2017.

[13] Y. Zhang, W. Xiao, "Keyphrase generation based on deep seq2seq model", *IEEE Access*, 6:46047–46057, 2018.

[14] K. Cho, B. Merri¨nboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation", *In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, 2014.

[15] I. Sutskever, O. Vinyals, Q. V. Le, "Sequence to sequence learning with neural networks", *Advances in Neural Information Processing Systems*, 27, pages 3104–3112, 2014.

[16] D. Bahdanau, K. Cho, Y. Bengio, "Neural machine translation by jointly learning to align and translate", *CoRR, abs/1409.0473*, 2014.

[17] S. Chopra, M. Auli, A. M. Rush, "Abstractive sentence summarization with attentive recurrent neural networks", *In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, 2016.

[18] A. See, P. J. Liu, C. D. Manning, "Get to the point: Summarization with pointergenerator networks", *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1073–1083, 2017.

[19] E. Papagiannopoulou, G. Tsoumakas, "A review of keyphrase extraction", *CoRR, abs/1905.05044*, 2019.

[20] K. S. Hasan, V. Ng, "Automatic keyphrase extraction: A survey of the state of the art", *In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1262–1273, 2014.

[21] S. Siddiqi, A. Sharan, "Keyword and keyphrase extraction techniques: a literature review", *International Journal of Computer Applications*, 109(2), 2015.

[22] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[23] J. R. Quinlan, "Bagging, boosting, and c4.s", *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, AAAI '96, Volume 1, pages 725–730, 1996.

[24] A. Hulth, "Improved automatic keyword extraction given more linguistic knowledge", *In Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223, 2003.

[25] C. Wu, M. Marchese, J. Jiang, A. Ivanyukovich, Y. Liang, "Machine learning-based keywords extraction for scientific literature", *Journal of Universal Computer Science*, 13(10):1471–1483, October 2007.

[26] R. Bhowmik, "Keyword extraction from abstracts and titles", *In IEEE SoutheastCon 2008*, pages 610–617, April 2008.

[27] S. Mao, A. Rosenfeld, T. Kanungo, "Document structure analysis algorithms: a literature survey", *In Document Recognition and Retrieval X*, pages 197–207, Santa Clara, California, USA, January 2003.

[28] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, "The weka data mining software: An update", *SIGKDD Explorations Newsletter*, 11(1):10–18, November 2009.

[29] M. Krapivin, A. Autayeu, M. Marchese, E. Blanzieri, N. Segata, "Keyphrases extraction from scientific documents", *The Role of Digital Libraries in a Time of Global Change*, 2010.

[30] J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kubler, S. Marinov, E. Marsi, "Maltparser: A language-independent system for data-driven dependency parsing", *Natural Language Engineering*, 13(2):95–135, 2007.

[31] C. Cortes, V. Vapnik, "Supportvector networks", *Machine Learning*, 20(3):273–297, September 1995.

[32] L. Breiman, "Random forests", *Machine Learning*, 45(1):5–32, October 2001.

[33] J. Villmow, M. Wrzalik, D. Krechel, "Automatic keyphrase extraction using recurrent neural networks", *Machine Learning and Data Mining in Pattern Recognition*, pages 210–217, 2018.

[34] R. Mihalcea, P. Tarau, "TextRank: Bringing order into texts", *In Proceedings of EMNLP-04 and the 2004 Conference on Empirical Methods in Natural Language Processing*, July 2004.

[35] S. Brin, L. Page, "The anatomy of a large-scale hypertextual web search engine", *Comput. Netw. ISDN Syst.*, 30(1–7):107–117, April 1998.

[36] S. D. Gollapalli, C. Caragea, "Extracting keyphrases from research papers using citation networks", *In Proceedings of the TwentyEighth AAAI Conference on Artificial Intelligence*, AAAI '14, pages 1629–1635, 2014.

[37] C. Florescu, C. Caragea, "Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents", *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1105–1115, 2017.

[38] Y. HaCohen-Kerner, "Automatic extraction of keywords from abstracts", *Knowledge-Based Intelligent Information and Engineering Systems*, pages 843–849, 2003.

[39] T. Jo, J. Lee, "Latent keyphrase extraction using deep belief networks", *International Journal of Fuzzy Logic and Intelligent Systems*, 15(3):153–158, 2015.

[40] G. Hinton, S. Osindero, Y. Teh, "A fast learning algorithm for deep belief nets", *Neural Computation*, 18(7):1527–1554, July 2006.

[41] Z. Liu, X. Chen, Y. Zheng, M. Sun, "Automatic keyphrase extraction by bridging vocabulary gap", *In Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, CoNLL '11, pages 135–144, Stroudsburg, PA, USA, 2011.

[42] T. Tomokiyo, M. Hurst, "A language model approach to keyphrase extraction", *In Proceedings of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, MWE '03, Stroudsburg, PA, USA, 2003.

[43] M. Vidyasagar, "Kullback-leibler divergence rate between probability distributions on sets of different cardinalities", *In 49th IEEE Conference on Decision and Control*, pages 948–953, December 2010.

[44] D. Mahata, J. Kuriakose, R. R. Shah, R. Zimmermann, "Key2Vec: Automatic Ranked Keyphrase Extraction from Scientific Articles using Phrase Embeddings", *In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Volume 2, pages 634–639, New Orleans, Louisiana, USA, June 2018.

[45] A. N. Langville, C. D. Meyer, "Deeper Inside PageRank", *Internet Mathematics*, 1:3, 335-380, January 2004.

[46] F. Boudin, "pke: an open source pythonbased keyphrase extraction toolkit", *In Proceedings of COLING 2016, the 26th International Confer-*

ence on Computational Linguistics: System Demonstrations, pages 69–73, Osaka, Japan, December 2016.

[47] F. M. Harper, J. A. Konstan, "The movielens datasets: History and context", *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, December 2015.

[48] E. Çano, M. Morisio, "Music mood dataset creation based on last.fm tags", *In Computer Science & Information Technology (CS & IT)*, pages 15–26, Vienna, Austria, May 2017.

[49] Z. Zajac, "Goodbooks-10k: a new dataset for book recommendations", *FastML*, 2017.

[50] E. Çano, M. Morisio, "Characterization of public datasets for recommender systems", *In 2015 IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 249–257, September 2015.

[51] T. D. Nguyen, M. Kan, "Keyphrase extraction in scientific publications", *In Proceedings of the 10th International Conference on Asian Digital Libraries: Looking Back 10 Years and Forging New Frontiers*, ICADL '07, pages 317–326, 2007.

[52] S. N. Kim, O. Medelyan, M. Kan, T. Baldwin, "Automatic keyphrase extraction from scientific articles", *In Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26, Uppsala, Sweden, July 2010.

[53] Q. Zhang, Y. Wang, Y. Gong, X. Huang, "Keyphrase extraction using deep recurrent neural networks on twitter", *In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 836–845, 2016.

[54] W. Yih, J. Goodman, V. R. Carvalho, "Finding advertising keywords on web pages", *In Proceedings of the 15th International Conference on World Wide Web*, WWW '06, pages 213–222, 2006.

[55] X. Yuan, T. Wang, R. Meng, K. Thaker, D. He, A. Trischler, "Generating diverse numbers of diverse keyphrases", *CoRR, abs/1810.05241*, 2018.

[56] M. Dredze, H. M. Wallach, D. Puller, F. Pereira, "Generating summary keywords for emails using topics", *In Proceedings of the 13th International Conference on Intelligent User Interfaces*, IUI '08, pages 199–206, New York, USA, 2008.

[57] M. Grineva, M. Grinev, D. Lizorkin, "Extracting key terms from noisy and multitheme documents", *In Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 661–670, New York, USA, 2009.

[58] K. M. Hammouda, D. N. Matute, M. S. Kamel, "Corephrase: Keyphrase extraction for document clustering", *Machine Learning and Data Mining in Pattern Recognition*, pages 265–274, 2005.

[59] E. Çano, O. Bojar, "Keyphrase generation: A text summarization struggle", *In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Volume 1, pages 666–672, Minneapolis, Minnesota, USA, June 2019.

[60] E. Çano, O. Bojar, "Efficiency Metrics for Data-Driven Models: A Text Summarization Case Study", *CoRR, abs/1909.06618*, 2019.

[61] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, Z. Su, "Arnetminer: Extraction and mining of academic social networks", *In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 990–998, New York, USA, 2008.

[62] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B. Hsu, K. Wang, "An overview of microsoft academic service (mas) and applications", *In Proceedings of the 24th International Conference on World Wide Web*, WWW '2015, pages 243–246, New York, USA, 2015.

[63] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, D. McClosky, "The Stanford CoreNLP natural language processing toolkit", *In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, USA, June, 2014.

[64] S. Hochreiter, J. Schmidhuber, "Long short-term memory", *Neural Computation*, 9(8):1735– 1780, November 1997.

[65] L. Marujo, W. Ling, I. Trancoso, C. Dyer, A. W. Black, A. Gershman, D. M. Matos, J. Neto, J. Carbonell, "Automatic keyword extraction on twitter", *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, Volume 2, pages 637–643, Beijing, China, July 2015.

[66] D. Dahlmeier, H. T. Ng, 'A beam-search decoder for grammatical error correction", *In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP–CoNLL '12, pages 568–578, Stroudsburg, PA, USA, 2012.

[67] J. Gu, Z. Lu, H. Li, V. O. K. Li, David, C. Christopher, V. Vapnik, "Incorporating copying mechanism in sequence-to-sequence learning", *In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Volume 1, pages 1631–1640, Berlin, Germany, August 2016.

[68] Y. Zhang, Y. Fang, X. Weidong, "Deep keyphrase generation with a convolutional sequence to sequence model", *In 2017 4th International Conference on Systems and Informatics*, pages 1477–1485, November 2017.

[69] Y. N. Dauphin, A. Fan, M. Auli, D. Grangier, "Language modeling with gated convolutional networks", *In Proceedings of the 34th International Conference on Machine Learning*, ICML '17, Volume 70, pages 933–941, 2017.

[70] Z. Tu, Z. Lu, Y. Liu, X. Liu, H. Li, "Modeling coverage for neural machine translation", *In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 76–85, 2016.

[71] J. Chen, X. Zhang, Y. Wu, Z. Yan, Z. Li, "Keyphrase generation with correlation constraints", *In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4057–4066, Brussels, Belgium, October-November 2018.

[72] C. L. A. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, I. MacKinnon, "Novelty and diversity in information retrieval evaluation", *In Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'08, pages 659–666, New york, USA, 2008.

[73] H. Ye, L. Wang, "Semi-supervised learning for neural keyphrase generation", *In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4142–4153, Brussels, Belgium, October-November 2018.

[74] W. Chen, H. P. Chan, P. Li, L. Bing, I. King, "An integrated approach for keyphrase generation via exploring the power of retrieval and extraction", *In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Volume 1, pages 2846–2856, Minneapolis, Minnesota, USA, June 2018.

[75] W. Chen, Y. Gao, J. Zhang, I. King, M. Lye, "Title-guided encoding for keyphrase generation", *CoRR, abs/1808.08575*, 2018.

[76] S. Misawa, Y. Miura, M. Taniguchi, T. Ohkuma, "Multiple keyphrase generation model with diversity", *Advances in Information Retrieval*, pages 869–876, 2019.

[77] J. Li, M. Galley, C. Brockett, J. Gao, B. Dolan, "A diversity-promoting objective function for neural conversation models", *In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California, USA, June 2016.

[78] Y. Wang, J. Li, H. P. Chan, I. King, M. R. Lyu, S. Shi, "Topic-aware neural keyphrase generation for social media language.", *CoRR abs/1906.03889*, 2019.

[79] H. P. Chan, W. Chen, L. Wang, I. King, "Neural keyphrase generation via reinforcement learning with adaptive rewards", *CoRR, abs/1906.04106*, June 2019.

[80] Y. Chen, M. Bansal, "Fast abstractive summarization with reinforce-selected sentence rewriting", *In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, Volume 1, pages 675–686, 2018.

[81] R. Paulus, C. Xiong, R. Socher, "A deep reinforced model for abstractive summarization", *CoRR, abs/1705.04304*, 2017.

[82] Y. Keneshloo, T. Shi, N. Ramakrishnan, C. K. Reddy, "Deep reinforcement learning for sequence to sequence models", *CoRR, abs/1805.09461*, 2018.

[83] M. Porter, "An algorithm for suffix stripping", *Program*, 40(3):211–218, 2006.

[84] R. Barzilay, M. Elhadad, "Using lexical chains for text summarization", *In Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, pages 10–17, 1997.

[85] S. Azzam, K. Humphreys, R. Gaizauskas, "Using coreference chains fortext summarization", *in Proceedings of the Workshop on Coreference and Its Applications, CorefApp '99*, pages 77–84, Stroudsburg, PA, USA. Association for Computational Linguistics.

[86] J. Goldstein, V. Mittal, J. Carbonell, M. Kantrowitz, "Multi-document summarization by sentence extraction", *In Proceedings of the 2000 NAACL-ANLP Workshop on Automatic Summarization*, NAACL-ANLP-AutoSum '00, Volume 4, pages 40–48, Stroudsburg, PA, USA, 2000.

[87] J. Fukumoto, "Multi-document summarization using document set type classification", *In Proceedings of the Fourth NTCIR Workshop on Research in Information Access Technologies Information Retrieval, Question Answering and Summarization*, NTCIR-4, Tokyo, Japan, June 2004.

[88] K. Wong, M. Wu, W. Li, "Extractive summarization using supervised and semisupervised learning", *In Proceedings of the 22Nd International Conference on Computational Linguistics*, COLING '08, Volume 1, pages 985–992, Stroudsburg, PA, USA, 2008.

[89] X. Wan, J. Yang, "Improved affinity graph based multi-document summarization", *In Proceedings of the Human Language Technology Conference of the NAACL* NAACL '06, pages 181–184, Stroudsburg, PA, USA, 2006.

[90] I. Mani, E. Bloedorn, "Multidocument summarization by graph search and matching", *In Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence*, AAAI '97/IAAI '97, 622–628, 1997.

[91] G. Erkan, D. R. Radev, , C. Christopher, V. Vapnik, "Lexrank: Graph-based lexical centrality as salience in text summarization", *Journal of Artificial Intelligence Research*, 22(1):457–479 December 2004.

[92] A. M. Rush. S. Chopra, J. Weston, "A neural attention model for abstractive sentence summarization", *In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, 2015.

[93] R. Nallapati, B. Zhou, C. Santos, C. Gulcehre, B. Xiang, "Abstractive text summarization using sequence-to-sequence rnns and beyond", *In Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, 2016.

[94] Y. Chen, M. Bansal, "Fast abstractive summarization with reinforce-selected sentence rewriting", *CoRR, abs/1805.11080*, May 2018.

[95] K. McKeown, D. R. Radev, "Generating summaries of multiple news articles", *In Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '95, pages 74–82, New York, USA, 1995.

[96] K. Kaikhah, "Automatic text summarization with neural networks", *2nd International IEEE Conference on Intelligent Systems*, Volume 1, pages 40–44, 2004.

[97] S. Harabagiu, F. Lacatusu, "Topic themes for multi-document summarization", *In Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 202–209, New York, USA, 2005.

[98] M. Grusky, M. Naaman, Y. Artzi, , "Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies", *In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 708–719, 2018.

[99] G. David, C. Christopher, V. Vapnik, "English gigaword", *Linguistic Data Consorium*, Philadelphia, USA, 2003.

[100] C. Lin, "Rouge: A package for automatic evaluation of summaries", *In Proceedings of ACL workshop on Text Summarization Branches Out*, page 10, 2004.

[101] K. Papieni, S. Roukos, T. Ward, W. J. Zhu, "Bleu: A method for automatic evaluation of machine translation", *In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002.

B Keyphrase Generation: A Text Summarization Struggle

Keyphrase Generation: A Text Summarization Struggle

Erion Çano
Institute of Formal and Applied Linguistics, Charles University, Prague, Czech Republic
cano@ufal.mff.cuni.cz

Ondřej Bojar
Institute of Formal and Applied Linguistics, Charles University, Prague, Czech Republic
bojar@ufal.mff.cuni.cz

Abstract

Authors' keyphrases assigned to scientific articles are essential for recognizing content and topic aspects. Most of the proposed supervised and unsupervised methods for keyphrase generation are unable to produce terms that are valuable but do not appear in the text. In this paper, we explore the possibility of considering the keyphrase string as an abstractive summary of the title and the abstract. First, we collect, process and release a large dataset of scientific paper metadata that contains 2.2 million records. Then we experiment with popular text summarization neural architectures. Despite using advanced deep learning models, large quantities of data and many days of computation, our systematic evaluation on four test datasets reveals that the explored text summarization methods could not produce better keyphrases than the simpler unsupervised methods, or the existing supervised ones.

1 Introduction

A valuable concept for searching and categorizing scientific papers in digital libraries is the *keyphrase* (we use *keyphrase* and *keyword* interchangeably), a short set of one or few words that represent concepts. Scientific articles are commonly annotated with keyphrases based on taxonomies of concepts and the authors' judgment. Finding keyphrases that best describe the contents of a document is thus essential and rewarding.

Most of the proposed keyphrase extraction solutions tend to be unsupervised (Florescu and Caragea, 2017; Nguyen and Luong, 2010; Rose et al., 2010; Bougouin et al., 2013; Campos et al., 2018) and generate terms by selecting the most appropriate candidates, ranking the candidates based on several features and finally returning the top N. Another way is to utilize datasets of texts and keywords for training supervised models with linguistic or other features to predict if candidates are keywords or not (Witten et al., 1999; Turney, 2000; Medelyan, 2009; Hulth, 2003).

All above methods propose N keyphrases for each article which are joined together with "," (or other separator like ";") to form the *keyphrase string* of that article. They suffer from various problems or discrepancies. First, they are unable to find an optimal value for N and require it as a preset parameter. Furthermore, semantic and syntactic properties of article phrases are analyzed separately. The meaning of paragraphs, sections or entire document is thus missed. Lastly, only phrases that do appear in the article are returned. Meng et al. (2017) recently proposed a deep supervised keyphrase generation solution trained on a big dataset. It successfully solves the last two problems above, but not the first one.

Motivated by recent advances in neural machine translation and abstractive text summarization (Vaswani et al., 2017; Foster et al., 2018; Rush et al., 2015; See et al., 2017), in this paper, we explore the possibility of considering keyphrase generation as an abstractive text summarization task. Instead of generating keywords one by one and linking them to form the keyphrase string, we consider the later as an abstractive summary of the concatenated paper title and abstract. Different recently-proposed text summarization architectures are tried on four test datasets of article keyphrases (Tanti et al., 2017; Rush et al., 2015; See et al., 2017). We trained them with a newly created dataset of 2.2 million article titles, abstracts and keyphrase strings that we processed and released.[1]

The selected text summarization models are compared with popular unsupervised and supervised methods using ROUGE (Lin, 2004) and full-match F_1 metrics. The results show that though

[1] http://hdl.handle.net/11234/1-2943

Proceedings of NAACL-HLT 2019, pages 666–672
Minneapolis, Minnesota, June 2 - June 7, 2019.

trained with large data quantities for many days, the tried text summarization methods could not produce better keywords than the existing supervised or deep supervised predictive models. In our opinion, a possible explanation for this is the fact that the title and the abstract may not carry sufficient topical information about the article, even when joined together. In contrast, when assigning keywords annotations of their paper, authors are highly influenced by the topic aspects of it.

This paper carries several contributions, despite the fact that no progressive result scores were reached. It is the first work that considers keyphrase generation as an abstractive text summarization task. We produced a large dataset of article titles, abstracts, and keywords that can be used for keyword generation, text summarization or similar purposes. Finally, we evaluated the performance of different neural network architectures on summarization of article keyword strings, comparing them with popular unsupervised methods.

2 Scientific Paper Datasets

Because of the open source and open data initiatives, many public datasets from various domains can be found online (Çano and Morisio, 2015). Among the several collections of scientific articles, some of them have gained considerable popularity in research literature. In Meng et al. (2017), we found a recent and big collection of 20K paper abstracts and keyphrases. These metadata belong to articles of computer science from ACM Digital Library, ScienceDirect, and Web of Science. In Hulth (2003), we found a collection of 2000 (1500 for train/val and 500 for testing) abstracts in English, together with titles and authors' keywords. The corresponding articles were published from 1998 to 2002 and belong to the discipline of *Information Technology*. Furthermore, Krapivin et al. (2010) released a dataset of 2000 (1600 for train/val and 400 for testing) full articles published by ACM from 2003 to 2005 in Computer Science domain. More information about similar keyphrase data collections or other available resources can be found in Hasan and Ng (2014) and in online repositories.[2]

Regarding text summarization, some of the most popular datasets are: DUC-2004 [3] mainly used for testing, English Gigaword (Napoles et al., 2012), CNN/Daily Mail described in Section 4.3 of (Nallapati et al., 2016) and Newsroom, a heterogeneous bundle of news articles described in Grusky et al. (2018). These datasets are frequently used for the task of predicting titles from abstracts or short stories. However, no keyphrases are provided; they do not serve to our purpose. ArnetMiner is a recent attempt to crawl scientific paper data from academic networks (Tang et al., 2008). The system extracts profiles of researchers from digital resources and integrates their data in a common network. A spin-off is the Open Academic Graph (OAG) data collection (Sinha et al., 2015).

Attribute	Train	Val	Test	Fullset
Records	2M	100K	100K	2.2M
Keyphrases	12M	575K	870K	13.4M
Title tokens	24M	1.3M	1.6M	27M
Abstract tokens	441M	21M	37M	499M
Av. Keyphrase	6	5.8	8.7	6.1
Av. Title	12.1	12.8	15.9	12.3
Av. Abstract	220	211	372	227

Table 1: Statistics of OAGK dataset

To produce a usable collection for our purpose, we started from OAG. We extracted *title*, *abstract* and *keywords*. The list of keywords was transformed into a comma-separated string and a language identifier was used to remove records that were not in English. Abstracts and titles were lowercased, and Stanford CoreNLP tokenizer was used for tokenizing. Short records of fewer than 20 tokens in the abstract, 2 tokens in the title and 2 tokens in the keywords were removed. For the test portion, we selected documents of at least 27, 3 and 2 tokens in each field. Data preprocessing stopped here for the release version (no symbol filtering), given that many researchers want to filter text in their own way. This new dataset named OAGK can be used for both text summarization (predicting title from abstract) and keyphrase extraction (unsupervised, supervised or deep supervised) tasks. Some rounded measures about each set of released data are presented in Table 1.

3 Keyphrase Extraction Strategies

3.1 Unsupervised and Supervised Methods

TopicRank is an extractive method that creates topic clusters using the graph of terms and phrases (Bougouin et al., 2013). Obtained topics are then ranked according to their importance in the document. Finally, keyphrases are extracted by pick-

[2] https://github.com/LIAAD/KeywordExtractor-Datasets

[3] https://duc.nist.gov/duc2004/

ing one candidate from each of the most important topics. A more recent, unsupervised and feature-based method for keyphrase extraction is YAKE! (Campos et al., 2018). It heuristically combines features like *casing*, *word position* or *word frequency* to generate an aggregate score for each phrase and uses it to select the best candidates.

One of the first supervised methods is KEA described by Witten et al. (1999). It extracts those candidate phrases from the document that have good chances to be keywords. Several features like *TF-IDF* are computed for each candidate phrase during training. In the end, Naïve Bayes algorithm is used to decide if a candidate is a keyword or not (binary classification). An improvement and generalization of KEA is MAUI (Medelyan, 2009). Additional features are computed, and bagged decision trees are used instead of Naïve Bayes. The author reports significant performance improvements in precision, recall and F_1 scores.

The above keyphrase extraction methods and others like Florescu and Caragea (2017) or Nguyen and Luong (2010) reveal various problems. First, they are not able to find an optimal value for N (number of keywords to generate for an article) based on article contents and require it as a preset parameter. Second, the semantic and syntactic properties of article phrases (considered as candidate keywords) are analyzed separately. The meaning of longer text units like paragraphs or entire abstract/paper is missed. Third, only phrases that do appear in the paper are returned. In practice, authors do often assign words that are not part of their article.

Meng et al. (2017) overcome the second and third problem using an encoder-decoder model (COPYRNN) with a bidirectional Gated Recurrent Unit (GRU) and a forward GRU with attention. They train it on a datasets of hundred thousands of samples, consisting of abstract-keyword (one keyword only) pairs. The model is entirely data-driven and can produce terms that may not appear in the document. It still produces one keyword at a time, requiring N (first problem) as parameter to create the full keyphrase string.

3.2 Text Summarization Methods

To overcome the three problems mentioned in Section 3.1, we explore abstractive text summarization models proposed in the literature, trained with article abstracts and titles as sources and keyword strings as targets. They are expected to learn and paraphrase over entire source text and produce a summary in the form of a keyphrase string with no need for extra parameters. They should also introduce new words that do not appear in the abstract. Two simple encoder-decoder variants based on LSTMs are described in Figure 3 of Tanti et al. (2017). MERGE (Figure 3.a) encodes input and the current summary independently and merges them in a joint representation which is later decoded to predict the next summary token. INJECT model (Figure 3.b) on the other hand injects the source document context representation to the encoding part of the current summary before the decoding operation is performed.

ABS is presented in Figure 3.a of Rush et al. (2015). The encoder (Figure 3.b) takes in the input text and a learned soft alignment between the input and the summary, producing the context vector. This soft alignment is the attention mechanism (Bahdanau et al., 2014). To generate the summary words, Rush et al. apply a beam-search decoder with a window of K candidate words in each position of the summary.

Pointer-Generator network (POINTCOV) depicted in Figure 3 of See et al. (2017) is similar to ABS. It is composed of an attention-based encoder that produces the context vector. The decoder is extended with a pointer-generator model that computes a probability p_{gen} from the context vector, the decoder states, and the decoder output. That probability is used as a switch to decide if the next word is to be generated or copied from the input. This model is thus a compromise between abstractive and extractive (copying words from input) models. Another extension is the coverage mechanism for avoiding word repetitions in the summary, a common problem of encoder-decoder summarizers (Tu et al., 2016).

4 Results

We performed experiments with the unsupervised and supervised methods of Section 3 on the first three datasets of Section 2 and on OAGK. All supervised methods were trained with the 2M records of OAGK train part. An exception was MAUI which could be trained on 25K records at most (memory limitation). In addition to the processing steps of Section 2, we further replaced digit symbols with # and limited source and tar-

Method	Hulth (500)		Krapivin (400)		Meng (20K)		OAGK (100K)	
	$F_1@5$	$F_1@7$	$F_1@5$	$F_1@7$	$F_1@5$	$F_1@7$	$F_1@5$	$F_1@7$
YAKE!	19.35	21.47	17.98	17.4	17.11	15.19	15.24	14.57
TOPICRANK	16.5	20.44	6.93	6.92	11.93	11.72	11.9	12.08
MAUI	20.11	20.56	23.17	23.04	22.3	19.63	19.58	18.42
COPYRNN	**29.2**	**33.6**	**30.2**	**25.2**	**32.8**	**25.5**	**33.06**	**31.92**
MERGE	6.85	6.86	4.92	4.93	8.75	8.76	11.12	13.39
INJECT	6.09	6.08	4.1	4.11	8.09	8.09	9.61	11.22
ABS	14.75	14.82	10.24	10.29	12.17	12.09	14.54	14.57
POINTCOV	22.19	21.55	19.87	20.03	20.45	20.89	22.72	21.49

Table 2: Full-match scores of predicted keyphrases by various methods

Method	Hulth (500)		Krapivin (400)		Meng (20K)		OAGK (100K)	
	R_1F_1	R_LF_1	R_1F_1	R_LF_1	R_1F_1	R_LF_1	R_1F_1	R_LF_1
YAKE!	37.48	24.83	26.19	18.57	26.47	17.36	20.38	14.54
TOPICRANK	32.0	20.36	14.08	11.47	21.68	15.94	17.46	13.28
MAUI	36.88	27.16	28.29	23.74	34.33	28.12	32.16	25.09
COPYRNN	**44.58**	**35.24**	**39.73**	**30.29**	**42.93**	34.62	**43.54**	**36.09**
MERGE	15.19	9.45	9.66	7.14	16.53	12.31	17.3	14.43
INJECT	14.15	8.81	9.58	6.79	15.6	11.21	14.3	11.08
ABS	27.54	19.48	25.59	18.2	28.31	22.16	29.05	25.77
POINTCOV	37.16	33.69	35.81	29.52	38.47	**35.06**	38.66	34.04

Table 3: Rouge scores of predicted keyphrases by various methods

get text lengths to 270 and 21 tokens, respectively. Vocabulary size was also limited to the 90K most frequent words.

The few parameters of the unsupervised methods (length and windows of candidate keyphrases for YAKE!, ranking strategy for TOPICRANK) were tuned using the validation part of each dataset. For the evaluation, we used F_1 score of full matches between predicted and authors' keywords. Given that the average number of keywords in the data is about 6, we computed F_1 scores on top 5 and top 7 returned keywords (**$F_1@5$, $F_1@7$**).

Before each comparison, both sets of terms were stemmed with Porter Stemmer and duplicates were removed. In the case of summarization models, keyphrases were extracted from their comma-separated summaries. We also computed ROUGE-1 and ROUGE-L F_1 scores ($\boldsymbol{R_1F_1}$, $\boldsymbol{R_LF_1}$) that are suitable for evaluating short summaries (Lin, 2004). The keywords obtained from the unsupervised methods were linked together to form the keyphrase string (assumed summary). This was later compared with the original keyphrase string of the authors.

Full-match results on each dataset are reported in Table 2. From the unsupervised models, we see that YAKE! is consistently better than TOPICRANK. The next two supervised models perform even better, with COPYRNN being discretely superior than MAUI.

Results of the four summarization models seem disappointing. MERGE and INJECT are the worst on every dataset, with highest score 13.39 %. Various predictions of these models are empty or very short, and some others contain long word repetitions which are discarded during evaluation. As a result, there are usually fewer than five predicted keyphrases. This explains why **$F_1@5$** and **$F_1@7$** scores are very close to each other.

ABS works slightly better reaching scores from 10.24 to 14.75 %. POINTCOV is the best of the text summarizers producing keyphrase predictions that are usually clean and concise with few repetitions. This is probably the merit of the coverage mechanism. There is still a considerable gap between POINTCOV and COPYRNN. Rouge-1 and Rouge-L F_1 scores are reported in Table 3. COPYRNN is still the best but POINTCOV is close. ABS scores are also comparable to those of MAUI and YAKE!. TOPICRANK, MERGE and INJECT are again the worst.

Regarding the test datasets, the highest result scores are achieved on Hulth and the lowest on Krapivin. We checked some samples of the later and observed that each of them contains separation tags (e.g., –T, –A, –B, Figure etc.) for indicating different parts of text in the original paper. A more intelligent text cleaning step may be required on those data.

5 Discussion

The results show that the tried text summarization models perform poorly on full-match keyword predictions. Their higher ROUGE scores further indicate that the problem is not entirely in the summarization process. Observing a few samples, we found differences between the two evaluation strategies. For example, suppose we have the predicted keyword *"intelligent system"* compared against authors' keyword *"system design"*. Full-match evaluation adds nothing to $\mathbf{F}_1\mathbf{@5}$ and $\mathbf{F}_1\mathbf{@7}$ scores. However, in the case of ROUGE evaluation, the prediction is partially right and a certain value is added to $\boldsymbol{R_1F_1}$ score. In follow up works, one solution to this discrepancy could be to try partial-match comparison scores like overlap coefficients.

Another detail that has some negative effect in full-match scores is keyword separation. The predicted string:

"health care,,,,immune system; human -; metabolism, immunity,,,,"

produces ["health care", "immune system", "human", "metabolism", "immunity"] as the list of keywords after removing the extra separators. Instead, we expected ["health care", "immune system", "human metabolism", "immunity"]. This again penalizes full-match scores but not $\boldsymbol{R_1F_1}$ score. A more intelligent keyword separation mechanism could thus help for higher full-match result scores.

A third reason could be the fact that we used the title and abstract of papers only. This is actually what most researchers do, as it is hard to find high quantities of article full texts for free. Article body is usually restricted. Abstractive summarization methods could still benefit from longer source texts. Using default hyperparameters for the models may have also influenced the results. Some parameter tuning could be beneficial, though.

The main reason could be even more fundamental. We trained abstractive summarization models on abstracts and titles with authors' keyphrases considered as golden ones. There might be two issues here. First, when setting their keywords, authors mostly consider the topical aspects of their work rather than paraphrasing over the contents. Abstracts and titles we used may not carry enough topical information about the article, even when joined together. Second, considering authors' keywords as golden ones may not be reasonable. One solution is to employ human experts and ask them to annotate each article based on what they read. This is however prohibitive when hundred thousands of samples are required. Extensive experiments on this issue may provide different facts and change the picture. For the moment, a safe way to go seems developing deep supervised generative models like the one of Meng et al. (2017) that predict one keyphrase at each step independently.

6 Conclusions

In this paper, we experimented with various unsupervised, supervised, deep supervised and abstractive text summarization models for predicting keyphrases of scientific articles. To the best of our knowledge, this is the first attempt that explores the possibility of conceiving article string of keywords as an abstractive summary of title and abstract. We collected and produced a large dataset of 2.2 million abstracts, titles and keyphrase strings from scientific papers available online. It can be used for future text summarization and keyphrase generation experiments. Systematic evaluation on four test datasets shows that the used summarization models could not produce better keywords than the supervised predictive models. Extensive experiments with more advanced summarizaiton methods and better parameter optimization may still reveal a different view of the situation.

Acknowledgments

This research work was supported by the project No. CZ.02.2.69/0.0/0.0/16_027/0008495 (International Mobility of Researchers at Charles University) of the Operational Programme Research, Development and Education, grant 19-26934X (NEUREM3) of the Czech Science Foundation and H2020-ICT-2018-2-825460 (ELITR) of the EU.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of the Sixth International Joint Conference on Natural Language*

Processing, pages 543–551. Asian Federation of Natural Language Processing.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, and Adam Jatowt. 2018. Yake! collection-independent automatic keyword extractor. In *Advances in Information Retrieval*, pages 806–810. Springer International Publishing.

Erion Çano and Maurizio Morisio. 2015. Characterization of public datasets for recommender systems. In *2015 IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 249–257.

Corina Florescu and Cornelia Caragea. 2017. Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1105–1115. Association for Computational Linguistics.

George Foster, Ashish Vaswani, Jakob Uszkoreit, Wolfgang Macherey, Lukasz Kaiser, Orhan Firat, Llion Jones, Noam Shazeer, Yonghui Wu, Ankur Bapna, Melvin Johnson, Mike Schuster, Zhifeng Chen, Macduff Hughes, Niki Parmar, and Mia Xu Chen. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *ACL (1)*, pages 76–86. Association for Computational Linguistics.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 708–719. Association for Computational Linguistics.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1262–1273.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223. Association for Computational Linguistics.

Mikalai Krapivin, Aliaksandr Autayeu, Maurizio Marchese, Enrico Blanzieri, and Nicola Segata. 2010. Keyphrases extraction from scientific documents. In *The Role of Digital Libraries in a Time of Global Change*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proc. ACL workshop on Text Summarization Branches Out*, page 10.

Olena Medelyan. 2009. *Human-competitive automatic topic indexing*. The University of Waikato, Phd Thesis.

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. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290. Association for Computational Linguistics.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX '12, pages 95–100, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thuy Dung Nguyen and Minh-Thang Luong. 2010. Wingnus: Keyphrase extraction utilizing document logical structure. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 166–169, Stroudsburg, PA, USA. Association for Computational Linguistics.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. In Michael W. Berry and Jacob Kogan, editors, *Text Mining. Applications and Theory*, pages 1–20. John Wiley and Sons, Ltd.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1073–1083. Association for Computational Linguistics.

Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June (Paul) Hsu, and Kuansan Wang. 2015. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 243–246, New York, NY, USA. ACM.

Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: Extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 990–998, New York, NY, USA. ACM.

Marc Tanti, Albert Gatt, and Kenneth P. Camilleri. 2017. What is the role of recurrent neural networks (rnns) in an image caption generator? *CoRR*, abs/1708.02043.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 76–85. Association for Computational Linguistics.

Peter D. Turney. 2000. Learning algorithms for keyphrase extraction. *Inf. Retr.*, 2(4):303–336.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. 1999. Kea: Practical automatic keyphrase extraction. In *Proceedings of the Fourth ACM Conference on Digital Libraries*, DL '99, pages 254–255, NY, USA. ACM.

C Efficiency Metrics for Data-Driven Models: A Text Summarization Case Study

Efficiency Metrics for Data-Driven Models: A Text Summarization Case Study

Erion Çano
Institute of Formal and Applied Linguistics, Charles University, Prague, Czech Republic
cano@ufal.mff.cuni.cz

Ondřej Bojar
Institute of Formal and Applied Linguistics, Charles University, Prague, Czech Republic
bojar@ufal.mff.cuni.cz

Abstract

Using data-driven models for solving text summarization or similar tasks has become very common in the last years. Yet most of the studies report basic accuracy scores only, and nothing is known about the ability of the proposed models to improve when trained on more data. In this paper, we define and propose three data efficiency metrics: data score efficiency, data time deficiency and overall data efficiency. We also propose a simple scheme that uses those metrics and apply it for a more comprehensive evaluation of popular methods on text summarization and title generation tasks. For the latter task, we process and release a huge collection of 35 million abstract-title pairs from scientific articles. Our results reveal that among the tested models, the Transformer is the most efficient on both tasks.

1 Introduction

Text summarization is the process of distilling the most noteworthy information in a document to produce an abridged version of it. This task is earning considerable interest, since shorter versions of long documents are easier to read and save us time. There are two basic ways to summarize texts. Extractive summarization selects the most relevant parts of the source document and combines them to generate the summary. In this case, the summary contains exact copies of words or phrases picked from the source. Abstractive summarization, on the other hand, paraphrases the information required for the summary instead of copying it verbatim. This is usually better, but also more complex and harder to achieve.

There has been a rapid progress in ATS (Abstractive Text Summarization) over the last years. The vanilla encoder-decoder with bidirectional LSTMs (Hochreiter and Schmidhuber, 1997) is now enhanced with advanced mechanisms like attention (Bahdanau et al., 2014) which has been widely embraced. It allows the model to focus on various parts of the input during the generation phase and was successfully used by Rush et al. (2015) to summarize news articles. Pointing (copying) is another mechanism that helps to alleviate the problem of unknown words (Gulcehre et al., 2016; Gu et al., 2016). Moreover, coverage (Tu et al., 2016) and intra-attention (Paulus et al., 2017) were proposed and utilized to avoid word repetitions, producing more readable summaries. RL (Reinforcement Learning) concepts like policy gradient (Rennie et al., 2017) were recently combined into the encoder-decoder architecture, alleviating other problems like train/test inconsistency and exposure bias (Paulus et al., 2017; Chen and Bansal, 2018).

All these developments helped to boost the ATS ROUGE (Lin, 2004) scores from about 30 % in Rush et al. (2015) to about 41 % in Paulus et al. (2017). This is an increase of roughly 37 % in the last three years. Yet all the studies evaluate the methods using datasets of a fixed size. Doing so they tell us nothing about the expected performance[1] of the models when trained with more data. Moreover, training time is rarely reported. We believe that this evaluation practice of data-driven models is incomplete and data efficiency metrics should be computed and reported.

In this paper, we propose three data efficiency metrics, namely *data score efficiency*, *data time deficiency* and *overall data efficiency*. The first two represent the output quality gain and the training time delay of the model per additional data samples. The third is the ratio between them and reflects the overall efficiency of the models w.r.t the training data. We also suggest a simple scheme that considers several values for each of the above metrics, together with the basic accuracy score, in-

[1] We use "performance" solely for the output quality, not the time needed to train the model or obtain the output.

Proceedings of The 12th International Conference on Natural Language Generation, pages 229–239,
Tokyo, Japan, 28 Oct - 1 Nov, 2019.

stead of reporting only the latter. The proposed scheme and the metrics can be used for a more detailed evaluation of supervised learning models.

Using them, we examine various recently proposed methods in two tasks: text summarization using the popular CNNDM (CNN/Daily Mail, Nallapati et al., 2016) dataset and title generation of scientific articles using OAGS, a novel dataset of abstract-title pairs that we processed and released.[2] According to our results, the best-performing and fastest methods in the two datasets are those of Paulus et al. (2017) and Chen and Bansal (2018). Regarding score and time efficiency, Transformer (Vaswani et al., 2017) is distinctly superior. In the future, we will examine the Transformer model on more data with different parameter setups. Applying our evaluation scheme to related tasks such as MT (Machine Translation) could also be beneficial.

Overall, this work brings the following main contributions: (i) We define and propose three data efficiency metrics and a simple evaluation scheme that uses them for a more comprehensive evaluation of data-driven learning methods. (ii) We use the scheme and metrics to benchmark some of the most recently proposed ATS methods and discuss their training times, ROUGE, and data efficiency scores. (iii) Finally, a huge collection of about 35 million scientific paper abstracts and titles is prepared and released to the community. To our best knowledge, this is the largest data collection prepared for title generation experiments.

2 Data Efficiency Metrics

2.1 Related Work

Training data efficiency of the data-driven learning models is little considered in the literature. An early work is that of Lawrence et al. (1998) who investigate the generalization ability of neural networks with respect to the complexity of the approximation function, the size of the network and the degree of noise in the training data. In the case of latter factor, they vary the size of the training data and the levels of Gaussian noise added to those data concluding that ensemble techniques are more immune to the increased noise levels. Performance variations w.r.t the training data sizes are not considered, though.

Al-Jarrah et al. (2015) review the research literature focusing in the computational and energy efficiency of the data-driven methods. They particularly consider data-intensive application areas (e.g., big data computing) and how sustainable data models can help for a maximal learning accuracy with minimal computational cost and efficient processing of large volumes of data.

Boom et al. (2016) examine a character-level RNN (Recurrent Neural Network) used to predict the next character of a text given the previous input characters. They assess the evolution of the network performance (in terms of perplexity) in four train and prediction scenarios as a function of the training time and input training sequences. According to their results, the efficiency of the model is considerably influenced by the chosen scenario.

A similar experiment is conducted by Riou et al. (2019) who explore reinforcement learning concepts on the task of neural language generation. They compare different implementations reporting not only performance scores, but also their evolution as a function of the cumulated learning cost and the training data size.

The most relevant work we found is the one by Hlynsson. et al. (2019) who propose an experimental protocol for comparing the data efficiency of a CNN (Convolution Neural Network) with that of HiGSFA (Hierarchical information-preserving Graph-based Slow Feature Analysis). They give an informal definition of data efficiency considering it as *performance as a function of training set size*. Three character recognition challenges are defined and the two methods are trained on increasing amounts of data samples reporting the corresponding accuracy scores.

2.2 Proposed Data Efficiency Metrics

Despite the experimental results and insights they bring, the above studies are still task and method specific. Moreover, their computation schemes are not generic or transferable and no formalization of the data efficiency is given. In this section, we define three novel and useful data efficiency metrics.

Suppose we train a data-driven method **M** on dataset **D** to solve task **T** and we test it based on performance score **S**. We also assume that the quality of the data samples in different intervals of **D** is homogeneous. In practice, this could be achieved by shuffling **D** before starting the experiments. For a certain training data size d, it takes t seconds to train the model m_d until convergence (i.e. until no further gains are observed with more

[2] http://hdl.handle.net/11234/1-3043

training time) and the score obtained by testing it on a standard and independent test dataset of a fixed size is s. We expect that for a certain increase Δd of training samples fed to **M**, it will require an extra time Δt to converge, and the resulting model $m_{d+\Delta d}$ will attain an extra Δs score. We can thus define and compute *data score efficiency* (score gain per additional data samples) Σ of method **M** as:

$$\Sigma \;=\; \Delta s \;/\; \Delta d \tag{1}$$

It is a measure of how smartly or effectively **M** interprets the extra data samples, or how well its performance score scales w.r.t the training data. Similarly, *data time deficiency* (the inverse of *data time efficiency*) Θ of **M** will be:

$$\Theta \;=\; \Delta t \;/\; \Delta d \tag{2}$$

This measures how slowly or lazily **M** interprets the additional samples.[3] Given two train and test runs (original and enlarged datasets) characterized by the above measures (training data: d, $d + \Delta d$; training times: t, $t + \Delta t$; achieved scores: s, $s + \Delta s$), we define the *overall data efficiency* E as:

$$E \;=\; \Sigma \;/\; \Theta \;=\; \Delta s \;/\; \Delta t \tag{3}$$

It is a measure of how smartly and quickly the models of **M** utilizes the data of **D** on task **T**.

In practice, using the absolute increments Δs, Δt, and Δd may produce small values of Σ which are hard to interpret and work with. Furthermore, Θ and E use training times which depend on the computing conditions (e.g., hardware setups). As a result, they are hardly reproducible across different computing environments. To overcome these limitations, we can instead use the relative increments $\Delta s/s$, $\Delta t/t$ and $\Delta d/d$, computing the corresponding *relative data efficiency metrics* as:

$$\sigma = \frac{\Delta s \;/\; s}{\Delta d \;/\; d} \tag{4}$$

$$\theta = \frac{\Delta t \;/\; t}{\Delta d \;/\; d} \tag{5}$$

$$\epsilon = \frac{\sigma}{\theta} = \frac{\Delta s \;/\; s}{\Delta t \;/\; t} \tag{6}$$

[3] Our *data time efficiency* (Δd / Δt) should not be confused with the *training throughput* as defined by Popel and Bojar (2018) for machine translation which reflects the time required for one model update given the additional data. Our Δt is the increase in the overall training time till convergence on the enlarged dataset in comparison with the original one.

These relative metrics and their values are practically easier to interpret and work with. Furthermore, they are transferable or reproducible in different computing setups which is important for cross-interpretation of the experimental results. We can express σ and θ values in percent and ϵ values as their ratio.

2.3 Assorted Remarks

The metrics presented above can be used to evaluate different data-driven methods or compare several parameter configurations of the same basic method (algorithm, neural network, etc.) and help us find the optimal one. In this sense, they are generic and task-independent. However, it is important to note that they do not represent "universal" or global attributes of method **M**. They are instead linear approximations that can give us local characterizations of **M** in certain intervals of **D**. In other words, high Σ (or σ) values of **M** in some intervals of **D** do not necessarily assure a decent generalization of **M**.

It is also important not to confuse the data efficiency with performance or quality. In our daily intuition, we often tend to consider highly efficient machines, techniques or methods as well-performing ones. Instead, according to the above definitions, a model can perform poorly but still be highly efficient w.r.t the training data. This happens if its performance scores on increasing training data cuts are all very low, but grow very quickly from one assessment to the next. A model can also yield high scores which grow very slowly on increasing data sizes (thus relatively small Σ and σ values). In this case it is a well-performing (maybe even the best) model on those data, but not a data efficient one.

From the data efficiency viewpoint, the best models would obviously be those of higher *data score efficiency* and lower *data time deficiency*, or higher *overall data efficiency*. In practice, performance is generally the most desired characteristic. As a result, *data score efficiency* values (Σ, σ or both) should be more important and worthy to report in most of the cases. Since models are trained only once, θ and ϵ should be less relevant. Nevertheless, they might be useful from a technical or theoretical perspective. They can be used for comparing different methods, comparing different parameter configurations of a method, or for trying run time optimizations.

3 A Comprehensive Evaluation Scheme

Since the sizes of the predictive models and the utilized datasets are consistently growing, it becomes more difficult and costly to use human expertise for the evaluation. The typical approach is to test automatically by means of standard datasets and scoring metrics which are popular. For example, in the case of text summarization task, it is very common to find evaluations of proposed methods using the full set of CNNDM only (Table 1 in Paulus et al., Table 3 in Lin et al., Table 1 in See et al., and more).

We believe there are serious shortcomings in this evaluation practice. Testing only one model of a method trained on a fixed-size data split does not reveal anything about its score trend when fed with more data. It thus becomes hard to discern the overall best method (out of a few that are compared) in a fair and objective way, especially if the achieved scores are similar. Moreover, training time is rarely reported and nothing is known about the time efficiency of the models.

To overcome the above limitations, we propose a more detailed evaluation scheme that considers accuracy scores together with the data efficiency metrics defined in Section 2.2. Again, suppose we have a dataset **D** of size d with homogeneous training samples, a standard performance score **S** and two methods **A** and **B** that we want to compare. The typical practice trains two single models a and b from **A** and **B** on entire d and reports accuracy scores s^a and s^b from the standard test set.

Instead, we suggest to split d in n equal parts of size d/n and form n intervals $d_1, d_2, \ldots, d_n$ of increasing sizes $d/n, 2d/n, \ldots, (n-1)d/n, d$. This way we can train $2n$ models $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$ on $d_1, d_2, \ldots, d_n$ and compute their scores $s^a_1, s^a_2, \ldots, s^a_n$ and $s^b_1, s^b_2, \ldots, s^b_n$ from the same test set. From Equation 4, we also compute $\sigma^a_1, \sigma^a_2, \ldots, \sigma^a_{n-1}$ using each two scores s^a_i and s^a_{i+1} of models a_i and a_{i+1}, together with $\sigma^b_1, \sigma^b_2, \ldots, \sigma^b_{n-1}$ from the **B** models.

We can now report up to $2n$ score values and $2(n-1)$ relative data score efficiency values. For conciseness, we can limit in s^a_n and s^b_n of the two biggest models. Also, given the local nature of the efficiency metrics, it make sense to report values from dispersed data intervals like the leftmost (σ^a_1 and σ^b_1), the middle ($\sigma^a_{n/2}$ and $\sigma^b_{n/2}$) and the rightmost (σ^a_{n-1} and σ^b_{n-1}) σ. The rightmost values

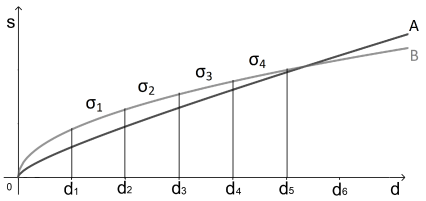

Figure 1: Illustration of the schema application

are probably more relevant for predicting the score trend on bigger training sizes. We can also compute and report the respective Σ values or even the θ and ϵ values in a similar fashion using the other equations of Section 2.2.

Getting back to **A** vs. **B**, we can first check s^a_n and s^b_n. If one of them is distinctly higher than the other, comparing the σ values may not be essential. The real worth comes when $s^a_n \approx s^b_n$, by contrasting the rightmost corresponding σ values (σ^a_{n-1} vs. σ^b_{n-1}). A significant difference of one against the other could suggest which of them will reach higher scores on a bigger training set.

To illustrate, we can see in Figure 1 two hypothetical graphs that approximate the variations of s^a and s^b over **D**. We have $n = 5$, training size $d = d_5$ and very similar performance scores ($s^a_5 \approx s^b_5$). Obviously, s^b grows faster than s^a till d_2, but then the situation is reversed, since $\sigma^a_3 > \sigma^b_3$ and $\sigma^a_4 > \sigma^b_4$. We can thus expect $s^a > s^b$ for $d > d_5$ which is what actually happens in this example ($s^a_6 > s^b_6$).

Using the traditional practice (computing s^a_5 and s^b_5 only) our verdict would be: ***A** and **B** perform (almost) the same on **D***. Instead, using the above scheme we can conclude that: ***A** and **B** perform (almost) the same on **D**, but **A** will probably perform better than **B** if trained on more data*. The scheme can be used to evaluate data-driven methods with different scores, on different tasks. In Section 5 we show the results we obtained by applying it to assess several advanced ATS methods.

4 Text Summarization Datasets

The tendency towards data-driven methods based on neural networks has encouraged experiments with large text collections for various tasks. In the case of ATS, one of the first big datasets was the annotated English Gigaword (Napoles et al., 2012; Rush et al., 2015) with over nine million news articles and headlines processed using CoreNLP of

	Split	Rec	SrcL	TgtL	Voc	Used
CNNDM	Train1	96K	784	54	380K	49K
	Train2	192K	780	57	555K	49K
	Train3	287K	786	55	690K	49K
	Valid	13K	769	61	–	–
	Test	11K	787	58	–	–
OAGS	Train1	500K	183	9	1.2M	98K
	Train2	1M	205	10	2.1M	98K
	Train3	1.5M	211	11	2.8M	98K
	Valid	10K	231	13	–	–
	Test	10K	237	12	–	–

Table 1: Statistics of used datasets. For each split, it shows the number of records (Rec), average length of source and target texts in tokens (SrcL, TgtL), total vocabulary size (Voc), and the number of most frequent words that were used (Used).

Manning et al. (2014). Each headline was paired with the first sentence of the corresponding article to create the training base for the experiments. DUC-2004 is another dataset[4], mostly used as an evaluation baseline, given its small size. It consists of 500 document-summary pairs curated by human experts. Newsroom is a recent and heterogeneous bundle of about 1.3 million news articles (Grusky et al., 2018).

CNNDM has become the most popular dataset for text summarization (Nallapati et al., 2016). It provides a large set of news articles and the corresponding multi-sentence summaries, unlike the three above that contain one-sentence summaries only. It is thus more suitable for training and testing summarization models of longer texts.

Title generation task, on the other hand, requires data samples of shorter texts and one-sentence titles. Collections of abstracts and titles from scientific articles are well suited for exploring it. KP20k is a collection of 20K records of scientific paper metadata (title, abstract and keywords) presented by Meng et al. (2017). The metadata belong to articles of computer science from ACM Digital Library, ScienceDirect, and Web of Science.

The demand for more and more data has motivated initiatives that mine research articles from academic networks. One of them is ArnetMiner, a system that extracts researcher profiles from the Web and integrates the data into a unified network (Tang et al., 2008). A byproduct of that work is the OAG (Open Academic Graph) collection (Sinha et al., 2015).

To produce a big title generation dataset for our experiments, we started from OAG. First, *abstract*, *title*, and *language* fields were extracted from each record where they were available. In many cases, abstract language did not match the *language* field. We ignored the latter and used a language identifier to remove records that were not in English. Duplicates were dropped and the texts were lowercased. Finally, Stanford CoreNLP tokenizer was used to split title and abstract texts. The resulting dataset (OAGS, released with this paper) contains about 35 million abstract-title pairs and can be used for title generation experiments.

We had a quick look at the content of OAGS and observed that most of the papers are from medicine. There are also many papers about social sciences, psychology, economics or engineering disciplines. Given its huge size and the topical richness, the value of OAGS is twofold: (i) It can be used to supplement existing datasets on title generation tasks when more training data are needed. (ii) It can be used for creating byproducts of specific scientific disciplines or domains.

5 Text Summarization Evaluation

In this section, we apply the relative metrics of Section 2.2 and the evaluation scheme of Section 3 to benchmark several advanced methods on text summarization of news articles and title generation of scientific papers. We first introduce the methods and their parameters, together with the experimental data. Later, we present and discuss the achieved accuracy and data efficiency scores.

5.1 Tested Summarization Methods

The ability of recurrent neural networks to represent and process variable-length sequences has created a tradition of applying them on sequence-to-sequence tasks such as ATS or MT. In the case of ATS, the goal is to process the source text producing a target text that is shorter but still meaningful and easy to read.

Rush et al. (2015) were probably the first to implement attention in a network dedicated to ATS. Their model (ABS in the following) uses an encoder that learns a soft alignment (attention) between the source and the target sequences producing the context vector. In the decoding phase, it uses a beam-search decoder (Dahlmeier and Ng, 2012) with a window of 10 candidate words in each target position. There are 256 and 128 dimensions in the hidden layer and word embedding layer respectively. The authors reported state-of-

[4] https://duc.nist.gov/duc2004/

the-art results in the DUC-2004 testing dataset.

See et al. (2017) proposed Pointer-Generator (PCOV), a model that implements an attention-based encoder for producing the context vector. The decoder is extended with a pointing/copying mechanism (Gulcehre et al., 2016; Gu et al., 2016) that is used in each step to compute a generation probability p_{gen} from the context vector, the decoder states, and the decoder output in that step. This generation probability is used as a switch to decide if the next word should be predicted or copied from the input. Another extension is the coverage mechanism (keeping track of decoder outputs) for avoiding word repetitions in the summary, a chronic problem of encoder-decoder summarizers (Tu et al., 2016). The method was implemented with word embeddings and hidden layer of sizes 128 and 256 respectively.

Lin et al. (2018) tried a partial use of convolutions in their model (GLOBEN) to avoid word repetitions and semantic irrelevance in the summaries. They couple the encoder with a convolutional gated unit which performs global encoding of the source context and uses it to filter certain n-gram features and refine the output of the encoder in each time step. GLOBEN is a very big network (about 68M parameters on CNNDM) with three layers in the encoder and other three in the decoder, each of 512 dimensions.

A taxonomy of the above (and more) sequence-to-sequence methods and added mechanisms can be found in Shi et al. (2018). Authors present a detailed review of problems and proposed solutions based on network structures, training strategies, and generation algorithms. Furthermore, they develop and release a library (NATS) that implements combinations of mechanisms like attention, pointing, and coverage, analyzing their effects in text summarization quality. NATS was implemented with the same network parameters as PCOV. Intra-decoder attention and weight sharing of embeddings were added in the decoder.

The introduction of the Transformer (TRANS) architecture that removes all recurrent or convolutional structures reduced computation cost and training time (Vaswani et al., 2017). Totally based on attention mechanism and primarily designed for MT, Transformer can also work for text summarization, since all it needs to do is to learn the alignments between the input (source) texts and the output (target) summaries. Positional encoding is added to word embeddings to preserve the order of the input and output sequences. TRANS is the biggest model we tried, with four layers in both encoder and decoder, 512 dimensions in each layer, including the embedding layers, 200K training steps and 8000 warm-up steps.

Two observed problems in the encoder-decoder framework are the *exposure bias* and *train/test inconsistency* (Keneshloo et al., 2018). To overcome them, RL ideas have been recently applied. Paulus et al. (2017) use intra-attention to focus on different parts of the encoded sequence. This way it is less likely for their model (PGRL) to attend to the same parts of input in different decoding steps, and thus fewer word repetitions should appear in the summaries. To optimize for ROUGE or similar discrete evaluation metrics, they implement self-critical policy gradient training with reward function, a RL mechanism introduced by Rennie et al. (2017). PGRL was used with encoder and decoder of 256 dimensions and word embeddings of 128 dimensions.

Aiming for speed, Chen and Bansal (2018) developed an extractive-abstractive text summarizer (FASTRL) with policy-based reinforcement. It first uses an extractor agent to pick the most salient sentences or phrases, instead of encoding the entire input sequence which can be long. It then uses an encoder-decoder abstractor to rewrite (compress) the sentences in parallel. Actor-Critic policy gradient with reward function (Bahdanau et al., 2016) joins together the extractor and abstractor networks. Same as most models above, FASTRL uses 256 and 128 dimensions for the recurrent layer and the word embeddings.

In every experiment, no pretraining of word embeddings was performed. They were learned during the training of each model. Adam optimizer (Kingma and Ba, 2014) was used with $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. We chose mini-batches of size 16 in most of the cases (8 for GLOBEN and TRANS to avoid memory errors). All experiments were conducted on two NVIDIA GTX 1080Ti GPUs.

5.2 Used Data

To cope with limited computing resources, we used up to 1.5M records in our OAGS experiments. We also picked $n = 3$ for the scheme of Section 3 and created three splits of 500K, 1M and 1.5M samples each, together with the three

		CNNDM					OAGS				
Authors	**Model**	P	R_1	R_2	R_L	T_t	P	R_1	R_2	R_L	T_t
Rush et al.	ABS1	15M	26.66	8.81	24.46	135032	22M	24.75	10.05	21.84	48595
	ABS2	15M	28.56	10.42	25.57	185549	22M	26.6	11.5	23.33	61729
	ABS3	15M	29.64	11.55	26.32	243549	22M	27.86	12.15	24.48	73038
See et al.	PCOV1	14M	36.97	15.19	33.84	113110	21M	34.4	17.67	27.55	**30551**
	PCOV2	14M	38.56	16.03	35.09	138175	21M	35.18	18.06	28.83	42723
	PCOV3	14M	39.41	16.77	36.31	163744	21M	35.86	18.51	29.42	56538
Shi et al.	NATS1	15M	36.92	14.56	32.88	98791	–	–	–	–	–
	NATS2	15M	38.25	15.89	34.02	179689	–	–	–	–	–
	NATS3	15M	39.11	17.2	35.66	261794	–	–	–	–	–
Lin et al.	GLOBEN1	68M	36.53	14.9	34.11	658924	–	–	–	–	–
	GLOBEN2	68M	37.82	16.13	35.46	785622	–	–	–	–	–
	GLOBEN3	68M	38.67	16.94	36.25	875817	–	–	–	–	–
Vaswani et al.	TRANS1	81M	32.38	10.47	29.43	518924	129M	30.29	13.1	24.34	251802
	TRANS2	81M	36.76	14.54	33.82	579149	129M	34.17	17.49	28.46	269665
	TRANS3	81M	38.24	16.33	35.28	611359	129M	37.06	**19.44**	30.51	278602
Chen et al.	FASTRL1	–	36.95	14.89	34.69	**19601**	–	–	–	–	–
	FASTRL2	–	39.18	16.17	36.15	30485	–	–	–	–	–
	FASTRL3	–	40.02	**17.52**	37.24	52775	–	–	–	–	–
Paulus et al.	PGRL1	–	38.16	14.17	36.24	68942	–	35.52	16.81	28.65	43726
	PGRL2	–	39.88	15.31	37.89	81529	–	36.9	18.44	30.22	55324
	PGRL3	–	**40.83**	15.68	**38.73**	107179	–	**38.05**	19.23	**31.16**	74983

Table 2: Parameters, ROUGE F_1 scores and training times for each method on the splits of the two datasets

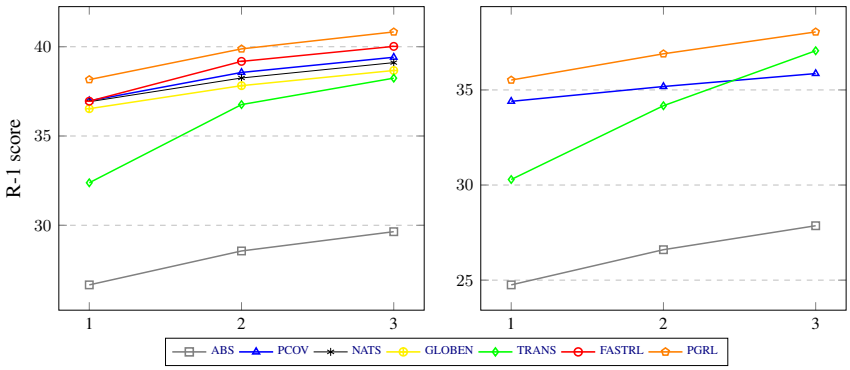

Figure 2: R_1 score trends of the three models of each method on CNNDM (left) and OAGS (right)

splits (one-third, two-thirds, and full) of CNNDM. Some statistics of the experimental data are shown in Table 1. Vocabulary sizes used in each experiment are shown in its last column.

The higher vocabulary sizes of OAGS splits cause a significant difference in parameters between the two corresponding models of each method. As we can see (Table 2), Transformer models grows from 81M in CNNDM to 129M in OAGS. Another difference between the two sets of experiments is in the maximal number of encoding and decoding steps (words in source and target texts). For CNNDM, we used 400 and 100 respectively. For OAGS, we chose 200 and 50, since paper abstracts and titles should not be longer.

5.3 Summarization Results

ROUGE scores and training times (in seconds) on CNNDM experiments are shown in the middle part of Table 2. The most accurate models are PGRL and FASTRL. They both implement policy-based training and optimize w.r.t ROUGE scores. The worst performer is ABS and the other four fall somewhere in between, reaching similar scores with each other.

The score differences between each third model and first one are usually small for all methods. We believe this has to do with the way ROUGE scores

		CNNDM						
Authors	**Models**	σ_1	σ_2	σ_L	θ	ϵ_1	ϵ_2	ϵ_L
Rush et al.	ABS12	7.13	18.27	4.54	37.41	0.19	0.45	0.12
	ABS23	7.64	21.92	5.93	63.18	0.12	0.35	0.094
See et al.	PCOV12	4.3	5.53	3.69	22.16	0.194	0.25	0.167
	PCOV23	4.46	9.33	7.03	37.4	0.119	0.249	0.188
Shi et al.	NATS12	3.6	9.13	3.47	81.89	0.044	0.112	0.042
	NATS23	4.53	16.6	9.74	92.35	0.049	0.18	0.106
Lin et al.	GLOBEN12	3.53	8.26	3.96	19.23	0.184	0.429	0.206
	GLOBEN23	4.54	10.15	4.50	23.2	0.196	0.437	0.194
Vaswani et al.	TRANS12	**13.53**	**38.87**	**14.92**	11.61	**1.166**	**3.34**	**1.285**
	TRANS23	8.14	24.88	8.72	**11.24**	0.724	2.214	0.776
Chen et al.	FASTRL12	6.04	8.6	4.21	55.53	0.109	0.155	0.067
	FASTRL23	4.33	16.87	6.09	147.78	0.029	0.114	0.041
Paulus et al.	PGRL12	4.51	8.05	4.55	18.26	0.247	0.441	0.249
	PGRL23	4.81	4.88	4.48	63.58	0.076	0.077	0.07

Table 3: Data efficiency scores of the models on CNNDM experiments. σ_X is computed based on the corresponding R_X score. Similarly, ϵ_X is computed based on σ_X and θ.

		OAGS						
Authors	**Models**	σ_1	σ_2	σ_L	θ	ϵ_1	ϵ_2	ϵ_L
Rush et al.	ABS12	7.47	14.43	6.82	27.03	0.277	0.534	0.252
	ABS23	9.47	11.3	9.86	36.64	0.259	0.309	0.269
See et al.	PCOV12	2.27	2.21	4.65	39.84	0.057	0.055	0.117
	PCOV23	3.87	5.04	4.09	64.67	0.06	0.077	0.063
Vaswani et al.	TRANS12	12.81	**33.51**	**16.93**	7.09	1.806	**4.724**	**2.386**
	TRANS23	**16.92**	22.3	14.41	**6.63**	**2.552**	3.364	2.173
Paulus et al.	PGRL12	3.89	9.7	5.48	26.52	0.146	0.366	0.207
	PGRL23	6.23	8.57	6.22	71.07	0.088	0.121	0.088

Table 4: Data efficiency scores of the models on OAGS experiments. σ_X is computed based on the corresponding R_X score. Similarly, ϵ_X is computed based on σ_X and θ.

are computed. A graphical representation of the R_1 trends for each method is depicted in Figure 2 (left). R_2 and R_L (not shown) behave similarly.

Results on OAGS are listed on the right side of Table 2. We could not run some of the models on OAGS data. The extractive part of FASTRL could not be easily adapted to perform word-level extraction of OAGS abstracts. Furthermore, NATS and GLOBEN ran out of memory very frequently. From the remaining four, PGRL is again the most accurate. TRANS follows and ABS is the weakest. R_1 score trends are shown in the Figure 2 (right).

Regarding training speed, on CNNDM we can see that FASTRL is absolutely the best, with a considerable difference from the second (PGRL). The slowest is GLOBEN with training times at least 17x higher than those of FASTRL. In fact, it took more than ten days to train GLOBEN on the full CNNDM data.

OAGS training times are lower than CNNDM ones, although OAGS data splits are 5.2 times bigger in number of training samples. This happens because OAGS source and target samples are actually much shorter. We see that PCOV is the fastest and TRANS is the slowest.

5.4 Efficiency Results

Using Equations 4, 5 and 6 of Section 2.2 we computed the relative efficiency metrics for every method. The values for CNNDM experiments are shown in Table 3. We see that TRANS is clearly the most efficient, with highest σ, lowest θ and highest ϵ. Its scores grow quickly (despite being relatively low) and training times grow slowly (despite being high) in both data intervals.

PCOV and GLOBEN manifest the slowest accuracy score gains (lowest σ), but GLOBEN comes second in time efficiency. NATS on the other hand, is very time inefficient, with highest θ and lowest ϵ. OAGS scores of Table 4 reflect a similar situation. TRANS leads and PCOV is again the worst. The values of the other two models appear somewhere in between.

It is not easy to explain the high score and time

efficiency of TRANS. GLOBEN is also time efficient but not score efficient. Both of them are the biggest (highest number of parameters) and deepest (many layers) networks we tried. The exclusive feature of TRANS is the lack of any recurrent structure. GLOBEN and the other five make use of at least one RNN in a certain phase. It is still hasty to infer that recurrent networks hinder score efficiency or that more attention boosts it.

An intuitive explanation could be the fact that in general, performance of deeper networks scales better with more data. It could also be that high-capacity networks are faster in interpreting large additions of training samples (thus low θ). In fact, using more layers and bigger training datasets is what has driven the progress of deep learning solutions in many application areas.

We plan to investigate this issue further in the future. One step could be to run more experiments on even bigger data sizes and smaller data intervals for checking at what point do accuracy scores keep growing. Transformer implementations with varying number of layers and other parameter setups can be further examined.

Investigating data efficiency of similar solutions to tasks like QA (Question Answering, Correia et al., 2018) with standard datasets such as SQuAD (Rajpurkar et al., 2018) could also be valuable.

6 Conclusions

In this paper, we defined three data efficiency metrics for a better evaluation of data-driven learning models. We also proposed a simple scheme for computing and reporting them, in addition to the basic accuracy scores. Text summarization and title generation tasks were chosen as a case study to see what insights the proposed scheme and metrics could reveal. For title generation, we also processed a dataset of about 35 million scientific titles and abstracts, released with this paper.

We applied seven recent ATS methods on the two tasks. According to our results, the two methods that mix RL concepts into the encoder-decoder framework are the fastest and the most accurate. A surprising result is the excellent efficiency of the popular Transformer model. As future work, we want to perform similar studies in analogous tasks like MT or QA. We would also like to investigate more in depth the Transformer model.

Acknowledgments

This research work was supported by the project No. CZ.02.2.69/0.0/0.0/16 027/0008495 (International Mobility of Researchers at Charles University) of the Operational Programme Research, Development and Education, the project no. 19-26934X (NEUREM3) of the Czech Science Foundation and ELITR (H2020-ICT-2018-2-825460) of the EU.

References

O. Y. Al-Jarrah, Paul D. Yoo, Sami Muhaidat, George K. Karagiannidis, and Kamal Taha. 2015. Efficient machine learning for big data: A review. *CoRR*, abs/1503.05296.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv e-prints*, abs/1607.07086.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Cedric De Boom, Sam Leroux, Steven Bohez, Pieter Simoens, Thomas Demeester, and Bart Dhoedt. 2016. Efficiency evaluation of character-level RNN training schedules. *CoRR*, abs/1605.02486.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686. Association for Computational Linguistics.

Alvaro Henrique Chaim Correia, Jorge Luiz Moreira Silva, Thiago de Castro Martins, and Fábio Gagliardi Cozman. 2018. A fully attention-based information retriever. *CoRR*, abs/1810.09580.

Daniel Dahlmeier and Hwee Tou Ng. 2012. A beam-search decoder for grammatical error correction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 568–578, Stroudsburg, PA, USA. Association for Computational Linguistics.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 708–719. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149, Berlin, Germany. Association for Computational Linguistics.

Hlynur Davíð Hlynsson., Alberto N. Escalante-B.., and Laurenz Wiskott. 2019. Measuring the data efficiency of deep learning methods. In *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods - Volume 1: ICPRAM,*, pages 691–698. INSTICC, SciTePress.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yaser Keneshloo, Tian Shi, Naren Ramakrishnan, and Chandan K. Reddy. 2018. Deep reinforcement learning for sequence to sequence models. *CoRR*, abs/1805.09461.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. Cite arxiv:1412.6980, Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Steve Lawrence, C Lee Giles, and Ah Chung Tsoi. 1998. What size neural network gives optimal generalization? convergence properties of backpropagation. Technical report.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proc. ACL workshop on Text Summarization Branches Out*, page 10.

Junyang Lin, Xu SUN, Shuming Ma, and Qi Su. 2018. Global encoding for abstractive summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 163–169. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

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. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290. Association for Computational Linguistics.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX '12, pages 95–100, Stroudsburg, PA, USA. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304.

Martin Popel and Ondřej Bojar. 2018. Training Tips for the Transformer Model. *The Prague Bulletin of Mathematical Linguistics*, 110(1):43–70.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1179–1195.

Matthieu Riou, Bassam Jabaian, Stéphane Huet, and Fabrice Lefèvre. 2019. Reinforcement adaptation of an attention-based neural natural language generator for spoken dialogue systems. *Dialogue & Discourse*, 10:1–19.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1073–1083. Association for Computational Linguistics.

Tian Shi, Yaser Keneshloo, Naren Ramakrishnan, and Chandan K. Reddy. 2018. Neural abstractive text summarization with sequence-to-sequence models. *CoRR*, abs/1812.02303.

Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June (Paul) Hsu, and Kuansan Wang.

2015. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 243–246, New York, NY, USA. ACM.

Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: Extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 990–998, New York, NY, USA. ACM.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 76–85. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

D Two Huge Title and Keyword Generation Corpora of Research Articles

Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 6663–6671
Marseille, 11–16 May 2020

Two Huge Title and Keyword Generation Corpora of Research Articles

Erion Çano, Ondřej Bojar
Institute of Formal and Applied Linguistics, Charles University
Prague, Czech Republic
{cano, bojar}@ufal.mff.cuni.cz

Abstract

Recent developments in sequence-to-sequence learning with neural networks have considerably improved the quality of automatically generated text summaries and document keywords, stipulating the need for even bigger training corpora. Metadata of research articles are usually easy to find online and can be used to perform research on various tasks. In this paper, we introduce two huge datasets for text summarization (OAGSX) and keyword generation (OAGKX) research, containing 34 million and 23 million records, respectively. The data were retrieved from the Open Academic Graph which is a network of research profiles and publications. We carefully processed each record and also tried several extractive and abstractive methods of both tasks to create performance baselines for other researchers. We further illustrate the performance of those methods previewing their outputs. In the near future, we would like to apply topic modeling on the two sets to derive subsets of research articles from more specific disciplines.

Keywords: text summarization, keyword generation, corpus construction, research articles, huge corpora

1. Introduction

The ongoing tendency towards data-driven solutions for more and more tasks such as MT (Machine Translation), TS (Text Summarization), KG (Keyword Generation), and other tasks related to natural languages has created incentives for crawling the Web to produce large text corpora of various types. Furthermore, recent open data initiatives of governments[1] and other institutions that encourage the publication of more data on the Web have induced the same effect. From academia, there are initiatives such as ArnetMiner (Tang et al., 2008) that try to integrate existing scientific data from various resources in common networks for easier retrieval and exploitation. Among the various types of texts published in the Web, the metadata of research articles (e.g., titles, abstracts, keywords, etc.) are probably the easiest to find in large quantities, since they are usually not restricted. In fact, small corpora of research articles were used since the 90s to explore extractive KG (Witten et al., 1999; Turney, 1999) and TS (Mani and Bloedorn, 1997; Goldstein et al., 2000) techniques.

Research on these tasks has switched from the extractive paradigm to the recent abstractive one that is based on sequence-to-sequence learning with neural networks. The respective models are usually data-hungry, emphasizing the need for larger corpora in both TS and KG tasks. In this paper, we first review the popular existing datasets used for TS and KG research. We later describe the processing steps we followed, starting from the retrieval of ArnetMiner OAG (Open Academic Graph) data collection to the creation of two novel and huge corpora: OAGSX[2] and OAGKX.[3] The first one contains more than 34 million records consisting of paper abstracts and titles. It is suitable for TS experiments (more specifically for title generation which is a form of TS). The second one contains roughly 23 million abstracts, titles, and lists of keywords and is best suited for KG experiments. The data samples in the two corpora were carefully examined and various statistics about the text lengths and the lexical similarities between abstracts, titles, and keywords are presented.

We also explored the performance scores (ROUGE for TS and F_1@k for KG) of existing extractive and abstractive TS and KG solutions, trying them on evaluation minisets derived from OAGSX and OAGKX. According to our results, the recent abstractive methods based on sequence-to-sequence learning take a considerable time to train but perform better than the extractive methods on both tasks. To the best of our knowledge, our two data collections are the biggest of their kind that can be found online for free. We release them under the Creative Commons By 4.0 License. As future work, we would like to perform topic recognition on the articles of the two collections. This may lead to the creation of many subsets of research articles data from more specific scientific disciplines.

2. Background

2.1. Text Summarization

Automatic TS research explores intelligent methods to compress text documents into shorter summaries that express the main ideas of the source. It is mostly driven by our need to have shorter and easy-to-read summaries of long documents for saving reading time. Sometimes, we also need to have summaries of conversation threads (e.g., emails or chat messages). Multi-document TS is important when we want concise information from a set of documents and summaries of conclusions from meetings (minuting) or other event discussions. Another type of summarization aims to create short client reviews about different aspects of certain products or services. Title generation is yet another form of TS which is about paraphrasing the content of a text to produce an appropriate title for it.

There are two fundamental approaches for performing TS. The extractive way tries to select the most important and relevant parts from the source document and combines them to produce a shorter summary which is concise, co-

[1] https://www.data.gov/open-gov
[2] http://hdl.handle.net/11234/1-3079
[3] http://hdl.handle.net/11234/1-3062

herent and readable. In this case, the target or output text contains verbatim copies of words or phrases taken from the source or input. The abstractive approach, on the other hand, learns to paraphrase the information required for the summary, instead of directly copying it from the source. This is somehow better, but the methodology is more complex and requires more resources. The TS research of the 90s and early 00s was mostly based on extractive methods. The respective techniques used unsupervised learning (Goldstein et al., 2000; Barzilay and Elhadad, 1997), supervised learning (Wong et al., 2008; Fukumoto, 2004) or graph methods (Erkan and Radev, 2004; Mani and Bloedorn, 1997) to select the most important lexical units from the source documents.

The abstractive approach has become popular in recent years, following the progress in sequence-to-sequence learning with neural networks (the encoder-decoder framework). LSTM neural networks (Hochreiter and Schmidhuber, 1997) are combined and enhanced with advanced mechanisms like the attention of Bahdanau et al. (2015) for a more effective learning of the alignments between the text sequences. Attention allows the model to focus on different segments of the input during generation and was successfully used by Rush et al. (2015) to summarize news articles. The problem of unknown words (not seen in source texts) was also mitigated by the copying technique (Gu et al., 2016; Gulcehre et al., 2016). Furthermore, the coverage (Tu et al., 2016a) and intra-attention (Paulus et al., 2017) mechanisms were proposed to alleviate word repetitions in the summaries, a notorious problem of the encoder-decoder models.

Scoring results were pushed even further just recently by mixing reinforcement learning concepts such as policy gradient (Rennie et al., 2017) into the encoder-decoder architecture. It optimizes the learning objective (higher summarization score) and still keeps an appropriate quality of the produced summaries. A recent performance comparison of various abstractive TS methods can be found in Çano and Bojar (2019a).

2.2. Keyphrase Generation

Keyphrase generation is the process of analyzing a document and producing sets of one or a few words (keywords or keyphrases, used interchangeably) that best represent its main concepts or topics. These keywords are frequently utilized nowadays to annotate digital objects (e.g., research articles, books, product descriptions, etc.) and quickly find them in digital libraries, online stores, etc. A keyword string is a concatenation of several keywords (commas or semicolons are typically used as separators) attached to one of those objects. The need to process large amounts of documents with missing keywords created incentives for research in automatic KG since the 90s. The popular supervised learning algorithms of that time were used by authors like Turney (2000) or Witten et al. (1999) in combination with lexical features to extract keywords from the documents. Furthermore, graph-based methods (Rose et al., 2010; Wan and Xiao, 2008) or other unsupervised KG methods (Campos et al., 2018; Nart and Tasso, 2014) were proposed later in the 00s.

The above extractive KG solutions were very successful because of their simplicity and execution speed. However, extractive KG suffers from a serious inherent handicap: its inability to produce absent keywords (keywords not appearing in the source text). Meng et al. (2017) analyzed the author's keywords in popular corpora. They observed that absent and present (keywords that also appear in the source text) keywords assigned by paper authors are almost equally frequent. It is thus a serious drawback to completely ignore the absent keywords.

The recent advances in language representation (Mikolov et al., 2013; Pennington et al., 2014) and sequence-to-sequence learning (Bahdanau et al., 2015; Vaswani et al., 2017) motivated several researchers like Meng et al. (2017) or Zhang and Xiao (2018) to explore abstractive KG in the context of the encoder-decoder framework. The encoder-decoder network structures were initially utilized to perform MT and got quick adoption on similar tasks like TS and KG that are also based on the sequence-to-sequence transformation between source and target texts. Furthermore, same as in TS research, various reinforcement learning concepts like adaptive rewards that are being explored are raising the performance scores even higher (Chan et al., 2019). Abstractive KG is now a vibrant research direction with more than a dozen of publications only in the last three years. More comprehensive surveys of KG literature can found in other recent publications like (Papagiannopoulou and Tsoumakas, 2019) and (Çano and Bojar, 2019b).

2.3. Scientific Article Data Sources

The current hype of deep neural networks has created strong incentives for producing data collections by crawling the web. The richest sets of language resources are used for machine translation (Resnik and Smith, 2003; Tiedemann, 2012; Mahata et al., 2016; Shi et al., 2005) and for sentiment analysis (Bosco et al., 2013; Çano and Bojar, 2019c; Maas et al., 2011; Çano and Morisio, 2019; Jiménez Zafra et al., 2015). They are mostly driven by the information technology giants that continuously improve their language-related applications and marketing companies to understand customers' perceptions about various online products.

TS and KG research of the 90s and early 00s was mostly based on extractive methods that did not rely on big training corpora. Things gradually changed in the late 00s with the rising popularity of the encoder-decoder framework. The current TS and KG methods are also highly dependent on the big language corpora since they are mainly based on sequence-to-sequence learning with neural networks. Some of the most popular corpora in TS and KG literature are presented in Table 1. One of the first big datasets was the annotated English Gigaword (Napoles et al., 2012) used for abstractive TS by Rush et al. (2015). It contains about nine million news articles and headline summaries. Each headline was paired with the first sentence of the corresponding article to create the training base for the experiments.

Newsroom (Grusky et al., 2018) is a very recent and heterogeneous bundle of about 1.3 million news articles. It contains writings published from 1998 to 2017 by 38 major newsrooms. Another recent dataset of news articles

Reference	Name	Content	# Docs
Napoles et al. (2012)	Gigaword	News	9 M
Grusky et al. (2018)	Newsroom	News	1.3 M
Nallapati et al. (2016)	CNN/DM	News	287 K
Hyperlink	DUC-2004	News	500
Hulth (2003)	Inspec	Papers	2000
Krapivin et al. (2010)	Krapivin	Papers	2304
Kim et al. (2010)	SemEval	Papers	244
Meng et al. (2017)	KP20k	Papers	567 K
Nikolov et al. (2018)	tit-gen	Papers	900 K
Nikolov et al. (2018)	abs-gen	Papers	5 M

Table 1: Summary and keyword generation datasets

is CNN/Dailymail of Nallapati et al. (2016). It has become the most popular corpus for text summarization experiments. This dataset provides a rich collection of news articles and the corresponding multi-sentence summaries (news highlights). It is thus very suitable for training and testing summarization models of longer texts. DUC-2004 is another dataset that was originally created for the Document Understanding Conference.[4] It has been mostly used as an evaluation baseline, given its small size. It consists of 500 document-summary pairs curated by human experts.

Besides using news articles, it is also possible to exploit texts of scientific articles for TS research. In fact, those kinds of texts have been used since long ago to conduct KG research. There are many relatively small datasets of scientific publications and the corresponding keywords that have been used for many years to test extractive or graph-based KG methods. One of the most popular KG datasets is Inspec released by Hulth (2003). It consists of 2000 paper titles (1500 for training and 500 for testing), abstracts and keywords from journals of Information Technology, published from 1998 to 2002.

Krapivin et al. (2010) released another collection of papers that has been frequently used in the literature. It consists of 2304 computer science papers published by ACM from 2003 to 2005. The advantage of this dataset is the availability of the full paper texts together with the corresponding metadata. A smaller dataset is SemEval of Kim et al. (2010) that was originally created for the Semantic Evaluation task. It contains 244 papers that belong to conference and workshop proceedings.

A few years ago, Meng et al. (2017) released KP20k which is today the most popular KG dataset. It contains 567830 Computer Science articles, 527830 used for training, 20 K for validation and 20 K for testing. This dataset has been used for training and comparing various recent abstractive KG methods. Nikolov et al. (2018) raised the data sizes even more by retrieving many scientific papers from libraries of biomedical research.[5] The authors derived and released two big (900 K and 5 M) corpora for TS (predicting abstracts from paper bodies) and title generation (predicting titles from abstracts).

Crawling public digital libraries or websites for text resources is an ongoing trend. ArnetMiner (Tang et al., 2008) is an initiative to integrate scientific data (publications, researcher profiles and more) from various resources in a common and unified network. A derivative product is the OAG data collection of scientific publications (Sinha et al., 2015). Each record is a JSON line with publication metadata like *authors*, *title*, *abstract*, *keywords*, *year* and more. In the following section, we describe the processing steps we performed on OAG collection to derive OAGSX and OAGKX datasets.

[4] https://duc.nist.gov/duc2004/

[5] https://www.nlm.nih.gov

Attribute	Title	Abstract
Total	449 M	6 B
Min / Max	3 / 25	50 / 400
Mean (Std)	13.1 (5.1)	182.2 (89.2)
Jindex	6.7 % (3.9 %)	
Overlap	77 % (18 %)	
Total size	34 408 509 title-abstracts	

Table 2: Token statistics of OAGSX

Attribute	Title	Abstract	Keywords
Total	290 M	4 B	270 M
Min / Max	3 / 25	50 / 400	2 / 60
Mean (Std)	12.8 (4.9)	175.1 (86.5)	11.9 (7.5)
Jindex	7.1 % (4 %)	6 % (4.8 %)	
Overlap	78 % (17 %)	68 % (25 %)	
Total size	22 674 436 title-abstract-keywords		

Table 3: Token statistics of OAGKX

3. OAGSX and OAGKX Corpora

For producing large TS and KG text collections, we utilized the text fields of the OAG bundle. From that same article set, we filtered the records containing at least the *title* and the *abstract* for OAGSX and those with the *title*, *abstract*, and *keywords* for OAGKX. We dropped the duplicate entries in each of our two collections. As a result, the samples inside each of the corpora are unique (there is still overlapping between OAGSX and OAGKX samples, since they were both derived from the OAG collection). An automatic language identifier[6] was used to remove the records with abstracts not in English. We also cleared the messy symbols and lowercased everything. Finally, Stanford CoreNLP (Manning et al., 2014) was used to tokenize the *title* and *abstract* texts.

After the preprocessing steps, we observed the size and token lengths of the records. Since there were many outliers (e.g., records with very long or very short abstracts), we removed all records with a title not in the range of 3-25 tokens and abstract not within 50-400 tokens. In the case of OAGKX, we also removed samples with keyword string not in the range of 2-60 tokens or 2-12 keywords. After this, OAGSX was reduced to a total of about 34.4 million records. OAGKX, on the other hand, shrank to about 22.6 million records.

[6] https://pypi.org/project/langid

Attribute	Value
Total	133 295 056
Min / Max	2 / 12
Mean (Std)	5.9 (3.1)
Present	52.7 % (28.3 %)
Absent	47.3 % (28.3 %)

Table 4: Keyword statistics of OAGKX

Some further statistics of the two final datasets are presented in Tables 2 and 3. In the case of OAGSX, the average title and abstract lengths are about 13.1 and 182.2 tokens respectively (standard deviation is always given in parenthesis). The corresponding values in OAGKX are 12.8 and 175.1 (slightly lower). For OAGKX we also see that the keyphrase strings contain 11.9 tokens on average. We also wanted to observe the lexical similarity between the titles and abstracts. One way for this is to compute the Jaccard similarity (Jindex in Tables 2 and 3) of the whole token sets using the following equation:

$$J(A,B) = \frac{|T \cap A|}{|T \cup A|} = \frac{|T \cap A|}{|T| + |A| - |T \cap A|} \quad (1)$$

where T is the set of unique tokens in the title and A is the set of unique tokens in the abstract. In OAGSX, the Jaccard similarity between abstracts and titles is 6.7 %. In OAGKX, it is 7.1 % between the abstracts and titles and 6 % between abstracts and keyword strings. Another indicator is the overlap $o(s,t) = \frac{|\{s\} \cap \{t\}|}{|\{t\}|}$ which represents the fraction of unique target tokens t (e.g., in the *title* or in the *keyword string* excluding punctuation symbols) that overlap with a source token (e.g., in *abstract*) s. The overlaps between titles and abstracts are very similar (77 % and 78 %) in both datasets. In the case of OAGKX, the overlap between abstracts and keyword strings is 68 %.

We further analyzed the keyword distribution in OAGKX (Table 4). There is a total of about 133 million keywords, with an average of 5.9 keywords per article. In abstractive KG experiments, it is also important to know the distribution of present and absent keywords. The present rate $p(s,k) = \frac{|k \cap s|}{|k|}$ is the fraction of the keywords k that also appear in the source text s. This is similar to the overlap, with the difference that there might be token repetitions within each counted keyword. The absent rate $a(s,k) = \frac{|k| - |k \cap s|}{|k|}$ is its complement or the fraction of keywords k that do not appear in the source text s. From Table 4 we see that the present and absent keywords in OAGKX are almost evenly distributed (52.7 % and 47.3 % each). This observation is in line with that of Meng et al. (2017), emphasizing once again the importance of the absent keywords.

Another interesting exploration we wanted to perform was the identification of the topics (or research domains) in each dataset record to report the corresponding statistics. This could lead to the creation of many subsets of OAGSX and OAGKX with scientific articles from more specific disciplines (clustering together the articles from the same research direction). Unfortunately, topic modeling was not easy to perform on OAGSX and OAGKX, given the huge size of the two corpora and our limited computational resources. It thus remains a potential future work.

We still inspected a few of the samples from each dataset manually. Their texts mostly belong to papers from biomedical disciplines but there are also papers about psychology, geology, or various technical directions. To our best knowledge, OAGSX and OAGKX are the largest available collections of scientific paper metadata that can be used for TS and KG experiments. Their importance is thus twofold: (i) They can supplement existing collections if more training samples are required. (ii) They can serve as sources for deriving article subsets of more specific scientific disciplines or domains.

4. Evaluation Experiments

We tried various extractive and abstractive methods for TS and KG on evaluation subsets from two corpora. In the following sections, we report the achieved performance scores of the automatic evaluation process. We also illustrate the output of each method with examples.

4.1. Title Generation

For the title generation experiments, we formed three evaluation subsets from OAGSX: a training set of 1 million samples, a validation set of 10 thousand samples and a test set of 10 thousand samples. To reduce the vocabulary size (important for abstractive text summarizers), we further replaced number patterns with the # symbol in each of them.

The most simple and raw baseline we used is Random-k (Random-1 in our case) which splits the source text into sentences and randomly picks k of them as its summary. In our case, since we are generating the title of the articles, we randomly pick only one of the abstract sentences as the predicted title. Random-1 can be considered as the lowest scoring boundary since it uses no intelligence at all.

Another popular baseline is Lead-k (Lead-1 in our case). It is based on the concept of "summary lead", which concisely explains the main idea of a text in its first sentence or first few sentences. Lead-1 picks the first sentence from the source text to generate its title.

LexRank is a stochastic graph-based method for assessing the importance of textual units in a source text (Erkan and Radev, 2004). When used to perform extractive TS, it computes the importance of those units using the concept of eigenvector centrality in the graph. The top k units (the top sentence in this case) are returned as the best summary of the document.

One of the abstractive text summarizers we used is PointCov of See et al. (2017) which is based on the encoder-decoder framework. In each decoding step, it implements the pointing/copying mechanism (Gu et al., 2016; Gulcehre et al., 2016) to compute a generation probability. The latter is used to decide whether the next word should be predicted or directly copied from the source sequence. Another feature is the implementation of the coverage mechanism (Tu et al., 2016a) which helps to avoid word repetitions in the target sequence. We trained PointCov with a hidden layer of 256 dimensions and word embeddings of 128 dimensions.

The other abstractive summarizer we picked is the Transformer model that represents one of the most important achievements in sequence-to-sequence learning of the last years (Vaswani et al., 2017). It is totally based on the attention mechanism, removing all recurrent or convolutional structures. Although it was primarily designed for MT, the Transformer can also work for text summarization. It basically learns the alignments between the input (source) texts and the output (target) summaries. As documented by Çano and Bojar (2019a), Transformer reveals the highest data efficiency scores on the popular TS datasets. We used the Transformer model with four layers in both encoder and decoder blocks, 512 dimensions in each layer, including the embedding layers, 200 K training steps, and 8000 warm-up steps. Both PointCov and Transformer were trained with Adam optimizer (Kingma and Ba, 2014) using $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ and mini-batches of 16 training samples. We used two NVIDIA GTX 1080Ti GPUs at once for the training process.

Random-1, Lead-1, and LexRank (the three extractive methods) were directly applied in the test set of 10 thousand examples. For PointCov and Transformer, we used all the three evaluation subsets. ROUGE-1, ROUGE-2, and ROUGE-L scores (Lin, 2004) were computed by comparing the title outputs of each method with the titles of the original papers. The results are presented in Table 5.

As we expected, Random-1 is the worst in all three ROUGE scores. Lead-1 performs well, reaching a peak score of 33.8 % in ROUGE-1. It is actually slightly better than LexRank in all the three metrics. Transformer and PointCov, which are the two abstractive neural networks we tried, perform better than the three extractive methods. They achieve similar results, but the Transformer leads with a peak score of 37.27 % in ROUGE-1. It is also important to note that the three extractive methods took only a few minutes to produce the outputs. PointCov and Transformer, on the other hand, required 3 – 4 days for the training.

An abstract, its author's title, and the titles predicted by the above five methods are illustrated in Table 6. As we can see, all the methods have generated titles that are longer than those of the authors. The titles of Lead-1 and LexRank are very similar, both based on the first sentence of the abstract. The transformer has produced a very long title with an unfinished sentence. This problem could be fixed by using a lower value for the length of the target text. PointCov has generated a shorter sentence than the Transformer, but it is not very coherent.

Method	R_1	R_2	R_L
Random-1	22.67	8.02	18.44
Lead-1	33.83	16.8	28.14
LexRank	29.4	12.83	24.03
PointCov	36.12	18.88	30.21
Transformer	37.27	19.12	30.78

Table 5: Results on OAGSX

Abstract: the central bank 's lender of last resort role was developed by a series of authors in the very late eighteenth and through the nineteenth centuries . it was tested in practice in a number of countries and was found to be effective in providing monetary stability in the face of adverse shocks . there have recently been attempts to broaden the role to make the central bank responsible for the stability of asset markets , or for protecting individual banks and there have recently also been claims that an international lender of last resort is necessary . this article considers and rejects these proposed extensions to the classic lender of last resort role
Author's title: the lender of last resort reconsidered
Random-1 title: this article considers and rejects these proposed extensions to the classic lender of last resort role
Lead-1 title: the central bank 's lender of last resort role was developed by a series of authors in the very late eighteenth and through the nineteenth centuries
LexRank title: the central bank 's lender of last resort role was developed in the late eighteenth and through the nineteenth centuries
PointCov title: the central bank 's lender and its implications for the stability of asset comparative analysis
Transformer title: the central bank 's lender of last resort role and its implications for the stability of asset markets : a comparative analysis of the central and in the lender of the

Table 6: KE scores on OAGKX

Method	F_1@5	F_1@7	F_1@10
TopicRank	17.12	20.81	20.75
RAKE	16.36	18.84	18.91
Maui	24.58	23.49	23.6
CopyRNN	28.15	28.93	28.96
CovRNN	27.76	29.15	29.04

Table 7: KE scores on OAGKX

4.2. Keyphrase Generation

We ran similar experiments on three evaluation sets derived from OAGKX: a training set of 631705 samples, a validation set of 10 thousand samples and a test set of 10 thousand samples. Once again, we tried and compared both extractive and abstractive KG methods. We used TopicRank of Bougouin et al. (2013) which is a popular graph-based extractive method that makes use of the PageRank algorithm (Brin and Page, 1998). It first uses clustering to group lexical units of the same topic. Then, it uses the graph-based ranking algorithm to score each topic cluster that is formed. At the end, one keyword is picked from each of the ranked clusters.

RAKE proposed by Rose et al. (2010) is one of the fastest available methods for extractive KG. It first removes punctuation symbols together with the stop words of the specified language and then creates a graph of word co-occurrences. Candidate words or phrases are scored based

Abstract: a complex polysaccharide accumulation was observed in the central nervous system (cns) of rats treated with d-penicillamine similar to lafora-like bodies . they have histochemical similarities comparable to bodies described in previous studies of lafora disease . the clinical usefulness of d-penicillamine has been limited by many side effects including renal damage . it is suggested that , in addition to d-penicillamine nephropathy , there are toxic effects of this drug on the cns
Title: polysaccharide accumulation in the central nervous system of d-penicillamine treated rats
Author's keywords: polysaccharide , central nervous system , side effect , d-penicillamine , lafora-like bodies , nephropathy
TopicRank keywords: central nervous system , d-penicillamine , accumulation , polysaccharide accumulation , cns , d-penicillamine effect , drug , lafora bodies , clinical , rats
RAKE keywords: clinical , polysaccharide , central nervous , rats , polysaccharide accumulation , lafora disease , renal damage , accumulation , lafora, cns
Maui keywords: central , central system , system , d-penicillamine , polysaccharide accumulation , polysaccharide , accumulation , lafora-like , lafora, bodies
CopyRNN keywords: central nervous , d-penicillamine , side effect , side , newborn rats , rat ileostomy , pregnant rats, nephropathy , mortality of rats , mortality
CovRNN keywords: side effect , central nervous , polysaccharide , rats , lafora bodies , polysaccharide effect , d-penicillamine , polysaccharide-derived , albino rats , trinitrobenzene sulfonic acid

Table 8: KE scores on OAGKX

on the degree and frequency of each word vertex in the graph. The k top-scoring candidates are returned as keywords. We also used Maui (Medelyan, 2009), a supervised extractive method that uses lexical features and bagged decision trees to predict whether a candidate phrase is a keyword or not.

CopyRNN (Meng et al., 2017) was the first abstractive KG method based on the encoder-decoder framework. Authors implemented the copying mechanism to balance between extracting present phrases from the source text with the generation of absent phrases. This work was followed by several recent studies that improve KG with various additional mechanisms. Finally, the last method we tried is CovRNN (Zhang and Xiao, 2018) which is very similar to CopyRNN. It tries to avoid the repeated keywords during generation by considering the correlation between the produced keywords at each generation step. This is achieved by implementing the coverage mechanism of Tu et al. (2016b).

We applied TopicRank and RAKE on the test set of 10 thousand records. Because of its memory limitations, Maui was trained on the first 30 thousand samples from the training set and tested on the test set. For CopyRNN and CovRNN, we used the full sizes of the three evaluation sets. For the comparison, we used F_1 scores of the full matches between the author's keywords and the top k keywords returned by each method. Given that each data sample has a variable-length keyword string, we picked the values 5, 7 and 10 for the k parameter. The obtained results are shown in Table 7. The first thing we can notice from the results is the fact that $F_1@7$ and $F_1@10$ scores are very similar to each other in each case. This is probably because few data samples contain more than 7 keywords in their keyword string (the average was 5.9). We also see that CopyRNN and CovRNN perform significantly better than the first three extractive methods. They achieve very similar scores in the three metrics. The peak score of 29.15 % is reached by CovRNN on $F_1@7$. From the three extractive methods, Maui performs better than the other two. TopicRank performs slightly better than RAKE. Once again, the training of the abstractive methods based on neural networks took about 3 days whereas the results of the extractive approaches (with the exception of Maui which was trained in few hours) were obtained in few minutes.

The source texts and the produced keywords (top ten) of a data sample are shown in Table 8. Apparently, both extractive and abstractive predictions are grammatically correct. However, few of the generations represent full keyword matches. There is also a considerable number of partial matches. The first four methods have produced certain word repetitions. We can also observe some "novel" (thou incorrect) phrases like "mortality of rats" or "trinitrobenzene sulfonic acid" that are produced by CopyRNN and CovRNN.

5. Conclusion

Today, we can find uncountable research article data that are freely available in digital libraries. Many relatively small collections of those data are frequently used to run text summarization and keyword generation experiments. In this paper, we described the steps we followed to process Open Academic Graph data and prepare two huge corpora: OAGSX of more than 34 million abstracts and titles that can be used for text summarization and OAGKX of about 23 million abstracts, titles, and keyword strings that can be used for keyword generation. To our best knowledge, these corpora of scientific paper metadata are the biggest freely available online. We also performed several experiments applying extractive and abstractive TS and KG methods on their subsets to help establish performance benchmarks that could be valuable to other researchers. In the future, we plan to apply topic modeling on the two collections for deriving many subsets of research articles from more specific scientific disciplines.

6. Acknowledgements

This research work was supported by the project No. CZ.02.2.69/0.0/0.0/16_027/0008495 (International Mobility of Researchers at Charles University) of the Operational Programme Research, Development and Education, the project no. 19-26934X (NEUREM3) of the Czech Science Foundation and ELITR (H2020-ICT-2018-2-825460) of the EU.

7. Bibliographical References

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Barzilay, R. and Elhadad, M. (1997). Using lexical chains for text summarization. In *In Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, pages 10–17.

Bosco, C., Patti, V., and Bolioli, A. (2013). Developing corpora for sentiment analysis: The case of irony and senti-tut. *IEEE Intelligent Systems*, 28(2):55–63, March.

Bougouin, A., Boudin, F., and Daille, B. (2013). Topicrank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 543–551. Asian Federation of Natural Language Processing.

Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, April.

Campos, R., Mangaravite, V., Pasquali, A., Jorge, A. M., Nunes, C., and Jatowt, A. (2018). Yake! collection-independent automatic keyword extractor. In Gabriella Pasi, et al., editors, *Advances in Information Retrieval*, pages 806–810. Springer International Publishing.

Çano, E. and Bojar, O. (2019a). Efficiency metrics for data-driven models: A text summarization case study. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 229–239, Tokyo, Japan, October–November. Association for Computational Linguistics.

Çano, E. and Bojar, O. (2019b). Keyphrase generation: A multi-aspect survey. In *Proceedings of the 25th Conference of Open Innovations Association FRUCT*, pages 85–94, Helsinki, Finland, Nov. 2019.

Çano, E. and Bojar, O. (2019c). Sentiment analysis of czech texts: An algorithmic survey. In *Proceedings of the 11th International Conference on Agents and Artificial Intelligence - Volume 2: NLPinAI*, pages 973–979, Prague, Czech Republic. INSTICC, SciTePress.

Çano, E. and Morisio, M. (2019). A data-driven neural network architecture for sentiment analysis. *Data Technologies and Applications*, 53(1):2–19.

Chan, H. P., Chen, W., Wang, L., and King, I. (2019). Neural keyphrase generation via reinforcement learning with adaptive rewards. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2163–2174, Florence, Italy, July. Association for Computational Linguistics.

Erkan, G. and Radev, D. R. (2004). LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Fukumoto, J. (2004). Multi-document summarization using document set type classification. In *Proceedings of the Fourth NTCIR Workshop on Research in Information Access Technologies Information Retrieval, Question Answering and Summarization, NTCIR-4, National Center of Sciences, Tokyo, Japan, June 2-4, 2004*.

Goldstein, J., Mittal, V., Carbonell, J., and Kantrowitz, M. (2000). Multi-document summarization by sentence extraction. In *Proceedings of the 2000 NAACL-ANLPWorkshop on Automatic Summarization - Volume 4*, NAACL-ANLP-AutoSum '00, pages 40–48, Stroudsburg, PA, USA. Association for Computational Linguistics.

Grusky, M., Naaman, M., and Artzi, Y. (2018). Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 708–719. Association for Computational Linguistics.

Gu, J., Lu, Z., Li, H., and Li, V. O. (2016). Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany, August. Association for Computational Linguistics.

Gulcehre, C., Ahn, S., Nallapati, R., Zhou, B., and Bengio, Y. (2016). Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149, Berlin, Germany, August. Association for Computational Linguistics.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780, Nov.

Hulth, A. (2003). Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223. Association for Computational Linguistics.

Jiménez Zafra, S. M., Berardi, G., Esuli, A., Marcheggiani, D., Martín-Valdivia, M. T., and Moreo Fernández, A. (2015). A multi-lingual annotated dataset for aspect-oriented opinion mining. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, Lisbon, Portugal.

Kim, S. N., Medelyan, O., Kan, M.-Y., and Baldwin, T. (2010). Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26. Association for Computational Linguistics.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. cite arxiv:1412.6980, Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Krapivin, M., Autayeu, A., Marchese, M., Blanzieri, E., and Segata, N. (2010). Keyphrases extraction from scientific documents. In Gobinda Chowdhury, et al., editors, *The Role of Digital Libraries in a Time of Global Change*.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Proc. ACL workshop on Text Summarization Branches Out*, page 10.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng,

A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June. Association for Computational Linguistics.

Mahata, S., Das, D., and Pal, S. (2016). WMT2016: A hybrid approach to bilingual document alignment. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 724–727, Berlin, Germany, August. Association for Computational Linguistics.

Mani, I. and Bloedorn, E. (1997). Multi-document summarization by graph search and matching. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence*, AAAI'97/IAAI'97, pages 622–628. AAAI Press.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Medelyan, O. (2009). *Human-competitive automatic topic indexing*. The University of Waikato, Phd Thesis.

Meng, R., Zhao, S., Han, S., He, D., Brusilovsky, P., and Chi, Y. (2017). Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 582–592. Association for Computational Linguistics.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv e-prints*, Jan.

Nallapati, R., Zhou, B., dos Santos, C., Gulcehre, C., and Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290. Association for Computational Linguistics.

Napoles, C., Gormley, M., and Van Durme, B. (2012). Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX '12, pages 95–100, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nart, D. D. and Tasso, C. (2014). A domain independent double layered approach to keyphrase generation. In *WEBIST*.

Nikolov, N. I., Pfeiffer, M., and Hahnloser, R. H. R. (2018). Data-driven summarization of scientific articles. *CoRR*, abs/1804.08875.

Papagiannopoulou, E. and Tsoumakas, G. (2019). A review of keyphrase extraction. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, n/a(n/a):e1339.

Paulus, R., Xiong, C., and Socher, R. (2017). A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Rennie, S. J., Marcheret, E., Mroueh, Y., Ross, J., and Goel, V. (2017). Self-critical sequence training for image captioning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1179–1195.

Resnik, P. and Smith, N. A. (2003). The web as a parallel corpus. *Computational Linguistics*, 29(3):349–380.

Rose, S., Engel, D., Cramer, N., and Cowley, W. (2010). Automatic keyword extraction from individual documents. In Michael W. Berry et al., editors, *Text Mining. Applications and Theory*, pages 1–20. John Wiley and Sons, Ltd.

Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389. Association for Computational Linguistics.

See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1073–1083. Association for Computational Linguistics.

Shi, L., Niu, C., Zhou, M., and Gao, J. (2005). A dom tree alignment model for mining parallel data from the web. In *COLING-ACL*, January.

Sinha, A., Shen, Z., Song, Y., Ma, H., Eide, D., Hsu, B.-J. P., and Wang, K. (2015). An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 243–246, New York, NY, USA. ACM.

Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., and Su, Z. (2008). Arnetminer: Extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 990–998, New York, NY, USA. ACM.

Tiedemann, J. (2012). Parallel data, tools and interfaces in opus. In Nicoletta Calzolari (Conference Chair), et al., editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may. European Language Resources Association (ELRA).

Tu, Z., Lu, Z., Liu, Y., Liu, X., and Li, H. (2016a). Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 76–85. Association for Computational Linguistics.

Tu, Z., Lu, Z., Liu, Y., Liu, X., and Li, H. (2016b). Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85, Berlin, Germany, August. Association for Computational Linguistics.

Turney, P. (1999). Learning to extract keyphrases from text.

Turney, P. D. (2000). Learning algorithms for keyphrase

extraction. *Inf. Retr.*, 2(4):303–336, May.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In I. Guyon, et al., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Wan, X. and Xiao, J. (2008). Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI'08, pages 855–860. AAAI Press.

Witten, I. H., Paynter, G. W., Frank, E., Gutwin, C., and Nevill-Manning, C. G. (1999). Kea: Practical automatic keyphrase extraction. In *Proceedings of the Fourth ACM Conference on Digital Libraries*, DL '99, pages 254–255, NY, USA. ACM.

Wong, K.-F., Wu, M., and Li, W. (2008). Extractive summarization using supervised and semi-supervised learning. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 985–992, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhang, Y. and Xiao, W. (2018). Keyphrase generation based on deep seq2seq model. *IEEE Access*, 6:46047–46057.