AUTH @ CLSciSumm 20, LaySumm 20, LongSumm 20

Alexios Gidiotis
[1] School of Informatics,
Aristotle University of Thessaloniki,
Thessaloniki, Greece
[2] Atypon Hellas,
Vasilissis Olgas 212,
Thessaloniki, Greece
gidiotis@csd.auth.gr

Stefanos Dimitrios Stefanidis
[1] School of Informatics,
Aristotle University of Thessaloniki,
Thessaloniki, Greece
[2] Atypon Hellas,
Vasilissis Olgas 212,
Thessaloniki, Greece
sstefanidis@atypon.gr

Grigorios Tsoumakas
[1] School of Informatics,
Aristotle University of Thessaloniki,
Thessaloniki, Greece
greg@csd.auth.gr

Abstract

We present the systems we submitted for the shared tasks of the Workshop on Scholarly Document Processing at EMNLP 2020. Our approaches to the tasks are focused on exploiting large Transformer models pre-trained on huge corpora and adapting them to the different shared tasks. For tasks 1A and 1B of CL-SciSumm we are using different variants of the BERT model to tackle the tasks of "cited text span" and "facet" identification. For the summarization tasks 2 of CL-SciSumm, LaySumm and LongSumm we make use of different variants of the PEGASUS model, with and without fine-tuning, adapted to the nuances of each one of those particular tasks.

1 Introduction

For scholars in every scientific domain, the ever growing amount of articles published each year has made the long-lasting challenge of keeping up with the recent literature significantly harder. In addition, there is an increasing need for making research accessible and relevant to the general public and not just a small group of researchers and practitioners. For example, taxpayers want to know where federal money supporting research goes. As a result, there is a need for different types of summaries that can either facilitate scientific research by compressing the key ideas discussed in a scientific paper or make scientific research more relevant for a lay audience.

It is obvious that tasking the author of a paper with writing multiple summaries of her work for different audiences is time-consuming. Consequently, the interest for methods that automatically summarize scientific documents in different styles and variations has increased significantly. Towards this direction, the 1st Scholarly Document Processing Shared Task (SDP 2020) (Chandrasekaran et al., 2020) introduces a number of different tasks that address these challenges. In addition to the original CL-SciSumm sub-tasks of previous years, the 2020 version includes tasks that are targeting the summarization of complete papers as well as the generation of lay summaries. The different tasks of SDP 2020 can be summarized as follows.

1. **CL-SciSumm:** The original CL-SciSumm challenge includes three sub-tasks. Given a set of reference papers (RP) and the corresponding papers that cite them (CP), task 1A requires for each citance (i.e. a sentence of the CP that references the RP) the identification of the "cited" text spans in the RP. In task 1B participants have to tag each cited span with the appropriate "facets" from a predefined set. Finally, for the optional task 2 a summary of the RP should be generated.

2. **LaySumm:** This task requires participants to generate a short summary for each given paper that accurately represents the content and at the same time is comprehensible and interesting to a lay audience.

3. **LongSumm:** In this task the requirement is to generate an extensive and detailed summary for each of the given scientific papers that sufficiently covers all the salient information.

Exploiting large language models that are pre-trained on huge corpora of unlabelled data and then adapting them to solve NLP problems has proven to be a very successful strategy. This type of approach has yielded state-of-the-art results in a variety of NLP tasks such as question answering, machine translation and summarization and has proven to be especially beneficial when the training data are limited. In this work our main focus is to explore large pre-trained Transformers like BERT (Devlin et al., 2018) and PEGASUS (Zhang et al., 2019)

Proceedings of the First Workshop on Scholarly Document Processing, pages 251–260
Online, November 19, 2020. ©2020 Association for Computational Linguistics
https://doi.org/10.18653/v1/P17

and how they can be effectively used in the context of the SDP 2020 shared task. For CL-SciSumm task 1A we fine-tune a BERT pairwise classifier that is able to identify "cited text spans" of the RP given a number of "citing spans" from multiple CPs. We further improve the efficiency of our system by adding a pre-filtering stage based on TF-IDF that selects good "candidates" for the BERT model. In task 1B we train a simple Logistic Regression classifier that uses embeddings from another pre-trained model, SciBERT (Beltagy et al., 2019), and is able to classify each cited text span into one of five distinct facets. We show that even an algorithm as simple as Logistic Regression can effectively learn a fairly complex task with very few training data when using features from a powerful pre-trained model such as SciBERT.

We approach task 2 of CL-SciSumm as well as LaySumm and LongSumm as abstractive summarization tasks. More specifically, we employ the PEGASUS pre-trained model that has demonstrated very good results in various abstractive summarization benchmarks. We use the pre-trained model without any additional training to generate a comprehensive summary of an article given the abstract as well as the information from the parts of the full text that are cited by other articles for task 2 of CL-SciSumm. For the LaySumm task, we fine-tune the PEGASUS model to compress and re-write the abstract of the given article in order to generate a summary suited for a lay audience. Finally, for the LongSumm task our approach makes use of the Divide-ANd-ConquER (DANCER) (Gidiotis and Tsoumakas, 2020) summarization method in combination with the PEGASUS model aiming to generate an accurate and detailed summary of the article by separately summarizing important sections of the full text.

The rest of this work is structured as follows. In section 2 we very briefly present different summarization approaches focusing on academic articles with emphasis on pre-trained Transformer models. In sections 3 to 5 we describe our approaches to each one of the three tasks and in section 6 we discuss our experiments and results.

2 Related Work

The task of summarizing scientific articles has received increased attention lately. Existing methods usually approach the problem in one of two fundamental ways. Extractive methods (Cohan and Goharian, 2015, 2018; Collins et al., 2017) focus mainly on identifying key sentences of the text and creating a summary by combining the extracted sentences. On the other hand, abstractive methods (Cohan et al., 2018; Subramanian et al., 2019; Zhang et al., 2019) are using language models conditioned on the input text in order to generate a summary.

Both extractive and abstractive methods when applied to scientific articles are typically taking as input the abstract and/or the full text of the article and try to generate an abstract-like summary. In contrast, DANCER (Gidiotis and Tsoumakas, 2020) learns to summarize different sections of the full text separately and combines the individual summaries into a single article summary.

Given the increased popularity and success of large pre-trained Transformer models in various NLP tasks, multiple approaches have decided to use similar models for summarization. Such approaches have either used pre-trained Transformers as encoders combined with a classification decoder that selects sentences in an extractive manner (Subramanian et al., 2019; Liu and Lapata, 2019) or have employed full encoder-decoder models that are pre-trained on various tasks and fine-tuned for abstractive summarization (Song et al., 2019; Dong et al., 2019; Yan et al., 2020).

One notable such model is the Pre-training with Extracted Gap-sentences for Abstractive SUmmarization Sequence-to-sequence (PEGASUS) (Zhang et al., 2019) model. PEGASUS is a Transformer encoder-decoder model pre-trained on massive corpora of documents (Web and news articles) that has demonstrated great potential on various summarization benchmarks. The pre-training of PEGASUS is based on optimizing the Gap Sentence Generation (GSG) objective where whole sentences of the input are masked and the model attempts to generate these gap-sentences from the rest of the input.

A number of summarization approaches we proposed for the CL-SciSumm task B in the previous years of the challenge, including extractive methods based on probabilistic models (Li et al., 2019) and large pre-trained BERT models (Zerva et al., 2019).

3 CL-SciSumm

3.1 Data Processing

The corpus of the CL-SciSumm shared task is split into two separate collections:

1. The manually annotated training set which consists of 40 articles and citing papers. For task 1A, we have multiple citances for each RP and each of these citances corresponds to a specific cited text span. For task 1B we are given the facet annotations for each one of the cited text spans and for task 2 human-written summaries are provided.
2. The ScisummNet corpus (Yasunaga et al., 2019), which consists of 1000 articles that are paired with multiple automatically annotated citing articles.

Out of the 40 articles in the manually annotated dataset, we randomly select 30 articles for the training set and 10 articles for the test set. For task 1A we created "positive" pairs of citing and reference spans as well as "negative" pairs of citing spans and randomly selected sentences from the RP. We also included the whole ScisummNet dataset into the training set of this task. For task 1B we are only able to use the manually annotated data since the ScisummNet dataset does not include facet annotations. One important notice about the task 1B data is the severe class imbalance which can potentially be problematic for the training of machine learning models.

The dataset for task 2 includes multiple human-written summaries for each article of the manually annotated dataset. Those summaries are annotated as "author summary", "community summary" and "human" in the JSON schema. We decided to use the summary labeled "human" as target summary because it was the one out of the three that was present in almost all articles of the dataset. In this task we will not be performing any additional training so we split the 40 articles into 20 for the validation set and 20 for the test set.

3.2 Task 1A

Our approach for task 1A makes use of the pre-trained BERT model (Devlin et al., 2018) and fine tunes it for the task. More specifically, we formulate the task as a sequence classification problem, where we are using the binary classification capabilities of the BERT architecture. Our main fine-tuning objective trains the model to take as input pairs of text spans and tries to predict if the second span is the corresponding span of the RP cited by the first span. We are training using "positive" and "negative" pairs with a 1:1 ratio. We found that creating the same number of negative pairs as the positive pairs yields the best results. The training set for this objective includes both the training part of the manually annotated dataset and the whole ScisummNet dataset.

During the prediction phase, our system evaluates each one of the given citing spans in a pairwise fashion with different sentences from the RP and selects at most two sentences that have the highest probability of being the corresponding cited text. If the probability difference between the top-2 sentences is higher than a threshold $T = 0.015$ then we only keep the first sentence. This way we are able to identify text spans instead of single sentences although we found that most of the time the cited text span is indeed a single sentence. To further improve the predictive power of our model we are providing additional context for the model by extending the both the citing and cited text spans with the previous and next sentence. When making predictions during the test phase we are using the citing span as is and only extend the candidate cited spans with the surrounding sentences.

Based on the findings by (Zerva et al., 2019) we also employed an additional pre-processing step before fine-tuning our model for the task specific objective. In our approach, we further pre-train BERT Base using the MLM objective on the ACL Anthology Reference Corpus (Bird et al., 2008).

In order to increase the computational efficiency of the pairwise evaluation, we are first using TF-IDF similarity to select the top-20 most similar sentences to the citing text span. Then we proceed on evaluating those "candidate" sentences with the pairwise model that we described previously.

3.3 Task 1B

For task 1B we decided to build a classification model that uses as input features contextual embeddings from the pre-trained SciBERT model (Beltagy et al., 2019) in order to classify each cited text into one of the five facets. Previous research has explored the use of contextual embeddings extracted from different layers of Transformer language models such as BERT (Devlin et al., 2018), ELMo (Peters et al., 2018) and GPT (Radford et al., 2019) as features for classification models. Also, (Ethayarajh, 2019) demonstrates some interesting insights of how the outputs of the different layers compare with each other. Here we decided to use the last layer of SciBERT to get the embeddings, because this model is more relevant to the domain of the task articles and we did not experiment with other model types. The contextual embeddings of each cited text have been obtained from the CLS vector of the last layer of SciBERT.

Although there are some occasions where multiple facets apply to the same span the vast majority of samples had a single facet. For this reason we decided to treat the task as a simple multiclass sequence classification problem. We experimented with multiple classification algorithms like Logistic Regression and Random Forests. We opted for simpler classification models due to the limited amount of training data that were severely limiting our ability to train more sophisticated models.

3.4 Task 2

We approach task 2 as an abstractive summarization problem. It has been suggested by (Yasunaga et al., 2019) that a combination of the abstract and the cited text spans is sufficient content to cover the main aspects and findings of an academic article. We follow this idea and propose a summarization scheme that takes those inputs and tries to generate a comprehensive summary of the article.

More specifically, we are using the PEGASUS model pre-trained on the arXiv dataset in order to generate the summary given the abstract and cited text spans identified from the previous tasks. Given that the combined input sequences are much longer than the maximum input size we can support for PEGASUS (due to GPU memory limitations) we decided to run the PEGASUS model twice, one for the abstract and the second for the cited texts and combine the two individual summaries into the final summary. The combination is a simple concatenation of the abstract summary with the cited text summary. Due to the small size of the manually annotated dataset we cannot expect to sufficiently train a PEGASUS model so we opted to use the model without any additional fine-tuning.

4 LaySumm

4.1 Data Processing

For the LaySumm task, the data are provided in the form of plain text files that have already been parsed from the paper PDFs. For each article in the dataset we are given three text files, one with the full text of the article, one with the abstract and one with the target lay summary. The corpus covers three distinct domains, namely epilepsy, archaeology, and materials engineering and consists of 573 articles in total. We split the dataset in three parts using 338 samples for the training set, 113 samples for the validation set and leaving 114 samples for the test set. We focused our pre-processing on cleaning noise and removing unwanted tokens and artifacts such as equations, tabular elements and references.

The PEGASUS model uses tokenization with the SentencePiece Unigram algorithm (Kudo, 2018) and required us to have all the text lowercased. The particular pre-trained model we are using comes with the Unigram 96k vocabulary that was created during the pre-training of the model. We identified that this vocabulary misses several symbols that appear quite frequently in the LaySumm data (e.g. Greek letters) so we decided to encode those symbols with other “complex” tokens from the vocabulary before tokenization in order for the model to be able to parse them. For example, the Greek letter α is replaced with the complex token “greekalpha” before being tokenized. This allows the model to successfully encode and learn the symbol and gives us the ability to backwards replace it to the original symbol during the decoding phase.

4.2 Lay Summary Generation

Our approach for the LaySumm task focuses on re-writing selected parts of the article in order to make them more relevant and easier to understand for the lay audience. Our main system uses the PEGASUS Large model and fine-tunes it on the task dataset. We are based on the idea that a lot of the key information that we want to include are present in the abstract of the article and we focus our methods to the task of re-writing and compressing the abstract. Our fine-tuning objective involves feeding the abstract as input to the PEGASUS model and using the provided lay summary as target for the summarization training.

We experimented with different variants of the pre-trained PEGASUS model. Those variants include: 1) the pre-trained PEGASUS model, 2) a model fine-tuned on the arXiv dataset and 3) a model fine-tuned on the PubMed dataset. All pre-trained models were open sourced by the authors of the PEGASUS paper. We further fine-tuned the different models on the specific LaySumm task using the provided dataset.

5 LongSumm

5.1 Data Processing

The abstractive dataset for the LongSumm corpus was given in the form of JSON files including article metadata, target summary and URLs to download the article PDFs. We used the provided scripts to download a total of 497 out of the 528 PDFs (we did not have access to the rest) and then extracted the abstract and section text from the downloaded files using Science-Parse[1]. We ended up with a total of 497 JSON files with the combined full text, abstract and target summaries. Out of these articles, 297 were used in the training set, 100 in the validation set and 100 in the test set.

The pre-processing steps followed in this dataset were similar to the ones we have described in the previous section and involve basic cleaning, normalisation and filtering operations. Again we use the same strategy for the tokens that are not supported by the PEGASUS vocabulary.

5.2 Long Article Summarization

Our approach for the LongSumm task is based on the Divide-ANd-ConquER (DANCER) summarization method which processes each section in a distributed way. The method uses text similarities between sentences of the summary and sections of full text in order to create better alignment during training and learns a summarization model that is able to summarize each section of the article separately.

More specifically, our system selects "types" of sections, namely the *introduction,* methods, *results and* conclusion, and uses the PEGASUS model to generate a summary for each section. The corresponding summaries are then concatenated to form the complete summary of the article. When training this system we use as input the full text of the section and as target the part of the summary that is most similar to that particular section.

We first use ROUGE-1 recall as a similarity metric in order to assign each sentence of the summary to one of the selected sections of the full text and then we group all the sentences assigned to each section to form the target summary corresponding to this section. The complete system architecture is shown in Figure 1.

[1] https://github.com/allenai/science-parse

section	**keywords**
introduction	introduction, case
literature	background, literature, related
methods	method(s), techniques, methodology
results	result(s), experimental, experiment(s)
conclusion	conclusion(s), concluding
acknowledgments	acknowledgments

Table 1: The different section types and the common keywords that are used in order to identify them using heuristics. If the header of a section includes any of the keywords associated with a specific section type it is assigned to that section type. Sections that can't be matched with any section type are ignored.

5.3 Section Tagger

In order to select the aforementioned types of sections we employ a classification model that classifies each section of a given article into one of six distinct categories (introduction, literature, methods, results, conclusion, acknowledgments). Based on our experiments we found that the combination of introduction, methods, results and conclusion gives us the best summaries overall.

This classifier has a single LSTM (Hochreiter and Schmidhuber, 1997) layer with additive attention (Bahdanau et al., 2015) and takes as input subword level BPEmb embeddings (Heinzerling and Strube, 2019). This model is trained on full text sections from the arXiv dataset. To train this model, we select sections of the corpus where the heading includes specific keywords that are characteristic of the section type. These keywords are shown in Table 1. We skip sections where the heading does not match this pattern. The model is trained to take as input the text of the section (without the heading) and tries to predict the section category.

6 Results and Discussion

6.1 Experimental Setup

In our experiments for task 1A we are using the Tensorflow implementation of BERT *Base* provided by huggingface [2]. After pre-training for 20k steps on the ACL corpus we proceed on fine-tuning for another 4k steps on the pairwise classification objective for task 1A. When we are building the TF-IDF model we are only based on the manually annotated

[2] https://github.com/huggingface/transformers

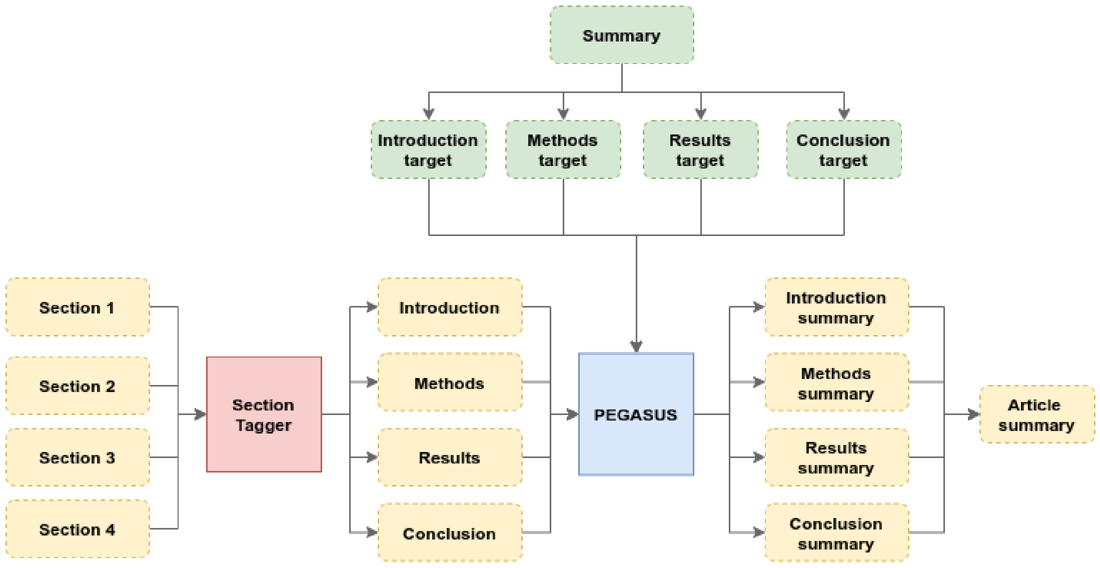

Figure 1: The main DANCER summarization framework combined with PEGASUS and the Section Tagger. During the training phase each section of the full text is paired with a part of the target. During the prediction phase important sections are selected with the help of the section tagger and each one is summarized separately. The individual summaries are then combined to form the article summary.

dataset. For task 1B we are using the SciBERT version open sourced by the authors of the original paper which is similar to the BERT Base model architecture. We are using the SciBERT model to get sentence level embeddings of size 768 which are then used as input to the classifiers.

All of our summarization methods are using the PEGASUS *Large* model which was pre-trained on the C4 and HugeNews dataset and was open sourced by the authors of the original paper. We also used two variations of this model that were fine-tuned for abstractive summarization. The first was fine-tuned for 74k steps on the arXiv dataset and the second for 100k steps on the PubMed dataset.

For task 2 of CL-SciSumm we are using the arXiv version of PEGASUS without any additional fine-tuning. We are running the model twice and generate summaries of up to 256 tokens for the abstract and the cited text spans identified from task 1A.

When fine-tuning our models on the LaySumm task we follow a very basic setup without extensive hyper-parameter tuning. More specifically, we used an input size of 1,024 tokens and an output size of 256 tokens since the evaluation scripts provided by the competition constrained the summary length to 150 words. We fine-tuned for 10k steps and monitor the ROUGE-1 F1 on the validation set in order to avoid overfitting. For the LongSumm task we are using the arXiv PEGASUS model and we further fine-tune it for 10k steps using the DANCER method on the dataset of the task. The hyper-parameters used are identical to the ones used for the LaySumm model. Detailed hyper parameters can be found in Appendix A.1.

	Macro			Micro		
Model	**P**	**R**	**F1**	**P**	**R**	**F1**
TF-IDF	14.64	10.45	12.20	15.23	9.30	11.55
BERT	**17.19**	**22.04**	**19.32**	**17.10**	**19.19**	**18.08**

Table 2: Results on our test set for task 1A. TF-IDF uses sentence similarities to select top-3 sentences. BERT is the proposed method.

6.2 Results

6.2.1 CL-SciSumm

We are evaluating task 1A on our test set (including only the manually annotated data) and measure the standard micro and macro precision, recall and F1 score. These metrics are shown in Table 2. For reference we are also comparing our method with a simple baseline that uses only TF-IDF to select the top-3 sentences for citing each span.

It can be seen clearly that the BERT based model is definitely superior to the baseline model and is able to correctly retrieve a fair amount of the cited text spans.

For task 1B we are evaluating our methods using 10-fold cross validation on the whole manually

Model	Precision	Recall	F1 score
Random Forest	40.71	34.98	33.84
Logistic Regression	**49.84**	**41.08**	**40.83**

Table 3: 10-fold cross validation results on the development set for task 1B. This results are independent of the outputs of task 1A.

	Macro			Micro		
Task	**P**	**R**	**F1**	**P**	**R**	**F1**
1A	13.47	18.20	15.48	14.03	17.49	15.57
1B	94.72	20.89	34.22	91.89	20.36	33.33

Table 4: Shared evaluation results on our test set using the best performing models for both tasks. Spans that are incorrectly retrieved in the first task are not being scored by the second task.

	F1			Recall		
Model	**R-1**	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
arXiv	47.93	25.36	31.66	46.13	23.85	30.17

Table 5: ROUGE scores of the proposed method for task 2 on the whole manually annotated dataset.

	F1			Recall		
Model	**R-1**	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
pre-trained	44.33	20.73	29.73	42.40	19.68	28.25
arXiv	**45.59**	20.68	29.84	**45.38**	20.53	29.59
PubMed	45.29	**21.26**	**30.29**	45.10	**21.03**	**29.92**

Table 6: Method comparison on our hold-out test set of LaySumm. Pre-trained is based on the original PEGASUS model while PubMed and arXiv are first fine-tuned on the PubMed and arXiv dataset respectively before additional fine-tuning on the LaySumm task.

	F1			Recall		
Model	**R-1**	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
pre-trained	41.14	16.01	25.16	37.75	14.57	23.04
arXiv	**44.56**	19.36	27.72	**42.98**	18.60	26.73
PubMed	44.25	**19.91**	**29.70**	42.06	**18.76**	**28.15**

Table 7: Method comparison on the blind test set of LaySumm.

annotated dataset. In Table 3 we show the macro precision, recall and F1 of our Logistic Regression classifier versus a Random Forest classifier.

These results show that very simple algorithms like Logistic Regression and Random Forests with SciBERT features perform well on this task with very few training examples. On the other hand, training more sophisticated models like neural networks was simply not feasible due to the small size of the dataset and the severe class imbalance. For example when we attempted to train neural networks for the task we ended up with models that only predicted the "methods" facet.

One should keep in mind that the previous results only measure the performance of the task 1B model, assuming that all "cited text spans" have been correctly identified by the task 1A model. We are also evaluating the combination of our best performing methods on our test set using the evaluation scripts provided by the competition. We use the text spans retrieved from task 1A as input for task 1B. The scores from the shared evaluation are shown in Table 4.

Finally, the evaluation of the summarization task 2 is done comparing the generated summary of each article with the "human" summary using ROUGE metrics (Lin, 2004). In Table 5 we present the results of our proposed approach on our test set. Those scores demonstrate the "zero-shot" capabilities of the PEGASUS pre-trained model which is able to perform well on a new task without any additional training.

6.2.2 LaySumm

When evaluating the results we used the official evaluation script provided by the competition which measures ROUGE-1, ROUGE-2 and ROUGE-L recall and F1-score. The results on our hold-out test set are shown in Table 6.

As expected, both of the fine-tuned models outperform the model without any prior fine-tuning since they are better adapted to summarizing academic articles. However, the differences between the two models are very small with the arXiv model achieving a better ROUGE-1 score while the PubMed one achieving better ROUGE-2 and ROUGE-L F1 scores.

In Table 7 we show the results on the blind test set of the competition. Similarly to the numbers on our own test set, the arXiv model performs slightly better in terms of ROUGE-1 while the PubMed model is better in terms of ROUGE-2 and ROUGE-L. Once again, both models have a clear advantage compared to the model without prior summarization fine-tuning.

6.2.3 LongSumm

Based on the LaySumm results the model fine-tuned on the arXiv dataset had superior performance to both the model without fine-tuning and the model fine-tuned on PubMed so we decided to use this variant for the LongSumm task.

In order to evaluate the impact of the section

	F1			Recall		
Model	**R-1**	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
notrain	24.64	6.18	16.29	**33.10**	8.01	**23.12**
arXiv	**25.81**	**8.09**	**18.01**	29.52	**8.91**	21.20
notrain-notag	24.47	4.72	15.07	28.03	7.95	21.48
arXiv-notag	24.26	4.58	15.15	25.53	4.68	16.28

Table 8: Section level comparison between methods on the LongSumm test set. Notrain uses the model fine-tuned on arXiv without additional training. ArXiv is additionally fine-tuned on LongSumm. Notrain-notag and ArXiv-notag are the same models but using heuristics instead of the section tagger for section selection.

	F1			Recall		
Model	**R-1**	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
notrain	41.88	10.66	17.46	**45.94**	11.42	**19.79**
arXiv	**43.52**	**12.12**	**18.67**	42.27	**11.59**	18.43
notrain-notag	30.97	6.94	14.45	26.38	5.67	12.49
arXiv-notag	31.36	7.47	15.40	25.75	5.91	13.10

Table 9: Article level comparison between methods on the test set of LongSumm.

	F1			Recall		
Model	**R-1**	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
notrain	49.91	14.23	19.19	**50.04**	**14.29**	**19.24**
arXiv	**50.11**	**15.37**	**19.59**	46.93	14.23	18.18
notrain-notag	38.89	10.65	17.12	31.32	8.54	13.64
arXiv-notag	38.27	9.48	16.93	29.20	7.14	12.79

Table 10: Article level comparison between methods on the blind test set of LongSumm.

tagger model we repeated the same experiments but this time instead of using the section tagger to help us select the appropriate sections we used the section headings and the heuristics described in 5.3.

First, we evaluated the performance at a section level using ROUGE-1, ROUGE-2 and ROUGE-L recall and f1-scores between the input section and the target section summary. Results for this evaluation are shown in Table 8. Second, we evaluate at an article level computing the same metrics between the full generated summary of each article and the full target summary. For this evaluation we use the official evaluation script provided by the competition and the results are shown in Table 9.

Looking at the section level results, we can see that fine-tuning the model with DANCER improves the results in every metric since it is better tuned for section level summarization compared to the model that is trained on whole articles. We should note that in this setup it is hard to have a direct comparison between the systems using the section tagger and the systems that use heuristics because using the section tagger results in a much larger test set.

The article level results can give us a better idea about the performance of the system on the LongSumm task itself. Here we run our summarization system to generate the section summaries, combine the summaries by concatenation to create the article level summary and compare it with the target article summary. The results on our test set show that once again DANCER training improves performance across the board. It is also clear that the section tagger has a very large effect as it improves both the trained and un-trained system by more than 10 ROUGE-1 points. This is clearly due to the fact that using the section tagger we include in the summary a lot more sections from the text that might not have a heading following the patterns from the heuristic approach.

Results on the test set of the competition are shown in Table 10. Similar to the results from our test set, we can see that the system trained with DANCER combined with the section tagger is clearly superior to all other systems.

7 Conclusion

We have presented the systems we developed for the SDP 2020 shared task. For task 1A we implemented an efficient pairwise classification approach based on the BERT model that tackles the “cited text identification” problem. For task 1B we show how a simple Logistic Regression classifier using pre-trained SciBERT embeddings as features can effectively learn to solve the problem of facet classification.

For the summarization tasks we employ different variants of the PEGASUS model and adapt them to the nuances of each particular task. For task 2 we use of the pre-trained PEGASUS model in a zero-shot setting to generate a summary given the abstract of an article along with the cited text spans. For LaySumm we propose a re-writing strategy based on the PEGASUS model that works on the abstract and generates a lay summary. Finally, we showcase how the PEGASUS model can be used to summarize an academic paper in a distributed way and we demonstrate an end-to-end system that generates a “long” summary by selecting key sections, summarizing each section independently and combining them to form the final summary.

References

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 2015 International Conference on Learning Representations*.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SCIBERT: A pretrained language model for scientific text. 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 3606–3611.

Steven Bird, Robert Dale, Bonnie J. Dorr, Bryan Gibson, Mark T. Joseph, Min Yen Kan, Dongwon Lee, Brett Powley, Dragomir R. Radev, and Yee Fan Tan. 2008. The ACL Anthology reference corpus: A reference dataset for bibliographic research in computational linguistics. In *Proceedings of the 2008 International Conference on Language Resources and Evaluation*.

M. K. Chandrasekaran, G. Feigenblat, Hovy. E., A. Ravichander, M. Shmueli-Scheuer, and A De Waard. 2020. Overview and Insights from Scientific Document Summarization Shared Tasks 2020: {CL-SciSumm}, {LaySumm} and {LongSumm}. In *Proceedings of the First Workshop on Scholarly Document Processing (SDP 2020)*.

Annan Cohan and Nazli Goharian. 2015. Scientific Article Summarization Using Citation-Context and Article Discourse Structure. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 390–400, Stroudsburg, PA, USA. Association for Computational Linguistics.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A Discourse-Aware Attention Model for Abstractive Summarization of Long Documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 615–621.

Arman Cohan and Nazli Goharian. 2018. Scientific document summarization via citation contextualization and scientific discourse. *International Journal on Digital Libraries*, 19(2-3):287–303.

Ed Collins, Isabelle Augenstein, and Sebastian Riedel. 2017. A Supervised Approach to Extractive Summarisation of Scientific Papers. In *Proceedings of the 2017 Conference on Computational Natural Language Learning*, pages 195–205.

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

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pages 13042–13054.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMO, and GPT-2 embeddings. 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 55–65.

Alexios Gidiotis and Grigorios Tsoumakas. 2020. A Divide-and-Conquer Approach to the Summarization of Long Documents. *arXiv preprint arXiv:2004.06190*.

Benjamin Heinzerling and Michael Strube. 2019. BPEMB: Tokenization-free pre-trained subword embeddings in 275 languages. In *Proceedings of the 2019 International Conference on Language Resources and Evaluation*.

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

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 2018 Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 66–75.

Lei Li, Yingqi Zhu, Yang Xie, Zuying Huang, Wei Liu, Xingyuan Li, and Yinan Liu. 2019. CIST@CLSciSumm-19: Automatic scientific paper summarization with citances and facets. In *CEUR Workshop Proceedings*, volume 2414, pages 196–207.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the 2004 Workshop on Text Summarization Branches Out, Post Conference Workshop of ACL*.

Yang Liu and Mirella Lapata. 2019. 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 3721–3731.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, volume 1.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI Blog*, 1(8).

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie Yan Liu. 2019. MASS: Masked sequence to sequence pre-training for language generation. In *Proceedings of the 2019 International Conference on Machine Learning*, pages 5926–5936.

Sandeep Subramanian, Raymond Li, Jonathan Pilault, and Christopher Pal. 2019. On Extractive and Abstractive Neural Document Summarizationwith Transformer Language Models. *arXiv preprint arXiv:1909.03186*.

Yu Yan, Weizhen Qi, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting Future N-gram for Sequence-to-Sequence Pre-training. *arXiv preprint arXiv:2001.04063*.

Michihiro Yasunaga, Jungo Kasai, Rui Zhang, Alexander R. Fabbri, Irene Li, Dan Friedman, and Dragomir R. Radev. 2019. ScisummNet: A Large Annotated Corpus and Content-Impact Models for Scientific Paper Summarization with Citation Networks. In *Proceedings of the 2019 AAAI Conference on Artificial Intelligence*, pages 7386–7393.

Chrysoula Zerva, Minh Quoc Nghiem, Nhung T.H. Nguyen, and Sophia Ananiadou. 2019. NaCTeM-UoM @ CL-SciSumm 2019. In *CEUR Workshop Proceedings*, volume 2414, pages 167–180.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J Liu. 2019. PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. *arXiv preprint arXiv:1912.08777*.

A Appendix

A.1 Model Hyper Parameters

Here we report detailed hyper parameters for the used for the training and evaluation of our models. For task 1A we pre-train on the ACL corpus for 20k steps with a batch size of 32 and a learning rate of 0.00003. Then we fine-tune on on the pairwise classification objective for another 4k steps with a batch size of 32 and a learning rate of 0.00001. For task 1B the Logistic Regression classifier was trained for 100 iterations with a C value of 0.1 and L1 regularization. Our Random Forest classifier has 100 estimators and uses the "Gini impurity" criterion.

The PEGASUS model used is an encoder-decoder model based on Transformers and has 16 Transformer blocks for the encoder and decoder with hidden size of 1,024 units, 16 self-attention heads and feed-forward layer size of 4,096 units. The model is pre-trained with MLM and GSG on a combination of the C4 and HugeNews datasets. For the LaySumm task the model is fine-tuned for 10k steps with a learning rate of 0.0001 and a batch size of 6 (mainly due to GPU memory constraints). For the LongSumm task the PEGASUS model is fine-tuned for 10k steps using the DANCER method, batch size of 6 and a learning rate of 0.0001.

A.2 Summarization Examples

In order to demonstrate the quality of the summaries generated by our methods, we present summaries of this paper generated by the arXiv and PubMed DANCER PEGASUS models.

DANCER arXiv: The 1st Scholarly Document Processing shared task (SDP 2020) is a new number of tasks that automatically summarize scientific documents in different styles and variations. In addition to the original CL-SciSumm sub-tasks of previous years, the 2020 version includes additional tasks that are targeting the summarization of complete papers as well as the generation of lay summaries. We present the systems we developed for the SDP 2020 shared task.

DANCER PubMed: For every scientific domain, the ever growing amount of articles published each year has made the long-lasting challenge of keeping up with the recent literature significantly harder. In addition to this, there is an increasing need for making research accessible and relevant to the general public and not just researchers and practitioners. For example, taxpayers want to know where federal money supporting research goes. There is a need for different types of summaries that can either facilitate scientific research compressing the key ideas discussed in a scientific paper or make scientific research relevant for a lay audience. We developed systems for a SDP 2020 shared task that use a pairwise approach to solve the cited text span identification problem, a pre-trained model to solve the problem of facet classification, and models to summarize academic papers.