CIST@CL-SciSumm 2020, LongSumm 2020: Automatic Scientific Document Summarization

Lei Li, Yang Xie, Wei Liu, Yinan Liu, Yafei Jiang, Siya Qi, and Xingyuan Li
Beijing University of Posts and Telecommunications (BUPT)
No.10 Xitucheng Road, Haidian District, Beijing, P.R.China
{leili,xieyang,thinkwee,lyinan,jiangyafei,qsy,lixingyuan}@bupt.edu.cn

Abstract

Our system participates in two shared tasks, CL-SciSumm 2020 and LongSumm 2020. In the CL-SciSumm shared task, based on our previous work, we apply more machine learning methods on position features and content features for facet classification in Task1B. And GCN is introduced in Task2 to perform extractive summarization. In the LongSumm shared task, we integrate both the extractive and abstractive summarization ways. Three methods were tested which are T5 Fine-tuning, DPPs Sampling, and GRU-GCN/GAT.

1 Introduction

The increasing scientific documents published on the Internet allow researchers to find more and more documents of interest. However, how to quickly and efficiently obtain the most important facts or ideas of a document is a big challenge. Summarization of scientific documents can mitigate this issue by presenting a brief summary of the whole document to researchers. This year, we participate in two shared tasks of SDP 2020 (Chandrasekaran et al., forthcoming). The CL-SciSumm shared task is the first medium-scale shared task on scientific document summarization in the field of Computational Linguistics and aims to generate a structured summary for the RP (Reference Paper) with the utilization of 10 or more CPs (Citing Papers). The LongSumm shared task opted to leverage blogs created by researchers in the NLP (Natural Language Processing) and Machine learning communities and use these summaries as reference summaries to generate the abstractive and extractive summaries for scientific papers.

In this paper, we will introduce our methods, experiments, and results of two shared tasks. For the CL-SciSumm shared task, based on our previous work (Li et al., 2019), we continue to leverage similarity calculation on multiple features to perform citation linkage in Task1A. In Task1B, we first extract position features and content features of RT (Reference Text) and CT (Citation Text), then apply different machine learning methods to classify the facet. In Task2, we apply DPPs (Determinantal Point Processes) and GCN (Graph Convolutional Network) to perform extractive summarization this time. As for the LongSumm shared task, we retain those extractive methods in the Task2 of the CL-SciSumm shared task as the basis for our summarization system. Furthermore, we also introduce the GAT (Graph Attention Network) and apply an abstractive summarization method based on fine-tuning.

2 Related work

The Task1A of CL-SciSumm is a citation linkage task. The most intuitive method is to calculate and compare the similarity between the CTS (Citation Text Spans) and every text span in RP (Reference Paper), and select the RT with the highest similarity as the result. There are many ways to calculate the similarity, not only traditional IDF and Jaccard similarity, but also Levenshtein distance (Yujian and Bo, 2007). The basic characteristics of words often play an important role in similarity calculating. As the size of the data set continues to grow, neural network language models such as Word2vec (Goldberg and Levy, 2014) and BERT (Devlin et al., 2018) that contain the semantic similarity information in word-level can make a huge improvement. But these word embedding methods will gradually smooth the difference between keywords in the process of calculating, so WMD (Kusner et al., 2015) was proposed to pay attention to the feature mapping between words. In addition to improvement in feature extraction, researchers have also proposed many new algorithms to process features, such as introducing CNN (Kim, 2014) (Dos Santos

Proceedings of the First Workshop on Scholarly Document Processing, pages 225–234

https://doi.org/10.18653/v1/P17

and Gatti, 2014) into the NLP field to make more complex judgments on feature vectors, or using MatchPyramid (Pang et al., 2016) to process the similarity comparison focusing on the similarity between words.

Task1B of CL-SciSumm is essentially a classification task. Classification methods are mainly divided into two parts: Rule-based methods and supervised machine learning methods. Traditional supervised machine learning methods like LR (Logistic Regression) (Park, 2013), Adaboost (Freund and Schapire, 1997) and XGBoost (Chen and Guestrin, 2016) can be easily applied for this task. Besides, the neural networks, such as TextCNN (Kim, 2014), TextRNN (Liu et al., 2016), TextRCNN (Lai et al., 2015), FastText (Joulin et al., 2016) and CharCNN (Zhang et al., 2015), can work directly on text, and generate dense vectors for classification.

The Task2 of CL-SciSumm and the LongSumm shared task are both summarization task. Recently, the research on automatic summarization tasks has mainly focused on two ways: extractive summarization and abstractive summarization. In the field of extractive summarization, We studied the sampling process used in DPPs (Kulesza and Taskar, 2012) where we calculated the kernel matrix using WMD sentence similarity for further sampling (Li et al., 2018). Zhong et al. (2019) explored how to make the system generate higher quality summaries. They selected three metrics: network architecture, knowledge transfer, and learning mode, and analyzed the impact of the three metrics on the quality of summary generation through experiments.

GCN is a powerful neural network framework processing graph structural data. Defferrard et al. (2016) extended the traditional CNN to non-Euclidean space and introduce local spectral filtering to optimize the propagation process during the training of the standard graph neural network. Kipf and Welling (2017) further studied the application of GCN in semi-supervised classification. GAT (Veličković et al., 2017) allocates different weights on different node neighbors to aggregate information. A document can also be converted into a graph. Yasunaga et al. (2017) introduced GCN in multi-document summarization. The clusters of documents were fed into RNN to obtain intermediate representations. Then GCN continued to extracting features considering the connections of documents clusters. At last, each sentence was scored based on its cluster-aware representations, and sentences with high score were chosen as summaries.

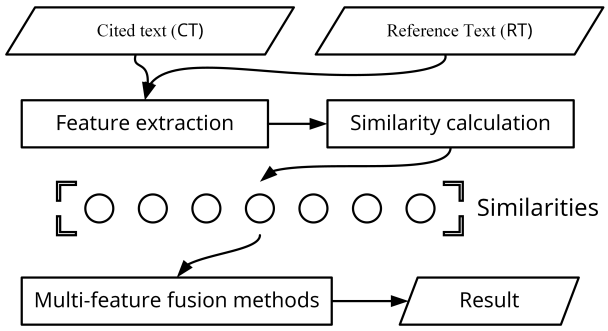

Figure 1: The complete process of Task1A.

As for abstractive summarization, Rush et al. (2015) introduced an attention mechanism to the Seq2Seq model, which enables the model to focus on words in specific positions in the original text via the weight matrix when generating abstracts, thus avoiding the problem of losing too much information due to long sentences. Since BERT (Devlin et al., 2018) has achieved great success in the field of NLP, the method of pre-training and fine-tuning has become a new paradigm. Researchers began to explore how to apply pre-trained models to natural language generation. At first, researchers tried to replace the encoder with a pre-trained BERT (Liu and Lapata, 2019), then more and more pre-training target functions for the Seq2Seq model were explored like masked generation (Song et al., 2019), denoising (Lewis et al., 2019), text-to-text (Raffel et al., 2019a). Some specially designed tasks for summarization have also been proposed, such as extracting gap-sentences (Zhang et al., 2019). We use the gap-sentence method in (Zhang et al., 2019) to combine and transform all the data, then utilize the T5 model (Lewis et al., 2019) to fine-tune and generate the summary.

3 Method

3.1 CL-SciSumm

3.1.1 Task1A

As shown in Figure 1, the citation linkage task, Task1A of CL-SciSumm, contains two steps: feature extraction and content linkage. In the feature extraction step, we perform similarity calculation based on different feature extraction ways for each RT and every CT (Citation Text) in CTS (Citation Text Spans), where some traditional features will be used, such as IDF similarity and Jaccard simi-

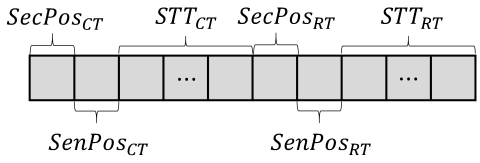

Figure 2: The position feature vector in Task1B.

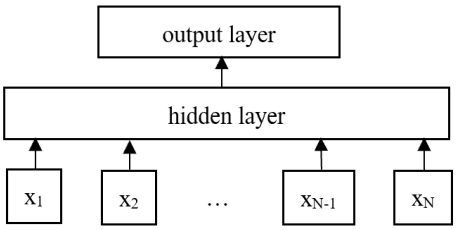

Figure 3: The architecture of FastText in Task 1B.

larity. Additionally sentence context information is used on the basis of these simple features in order to more comprehensively reflect the similarity information of the sentence. Besides, we also use the Lin and Jcn features of WordNet, word-cos, Word vector, and LDA-Jaccard (Li et al., 2019). LDA-Jaccard performs better than LDA on sparse topics, and it pays more attention to the union set of the same topic that both two sentences have. In the content linkage step, we add all the scores that each CT belonging to the same CTS, then sort all RTs by the final scores, and take the first N results as the final answer of Task1A. We use four multi-feature fusion methods: Voting-1.2, Voting-2.1, Jaccard-Focused, and Jaccard-Cascade based on our last year work (Li et al., 2019) by increasing the training set and adjusting the hyper-parameters.

3.1.2 Task1B

Our system applies multiple machine learning methods on multiple features representing different aspects of CT and RT. Since a scientific paper is well-structured and each section represents a different facet of the document, our first motivation is to leverage the position feature of CT and RT to classify which facet the citation belongs to. As shown in Figure 2, the position features are the relative positions of CT and RT, the relative positions of the sections that CT and RT belong to, and the section title text. Suppose the section id is sid, the total amount of sections is $tsid$, the sentence id is $ssid$, and the total amount of sentences is $tssid$. Then, the section relative position($SecPos$) of CT or RT is $sid/tsid$, and the sentence relative position($SenPos$) of CT or RT is $ssid/tssid$. Since the section title text(STT) of CT or RT also implicates the role it plays in the whole paper, we leverage TF-IDF to select the top 189 words as the keywords where each word occurs at least 3 times in the training set, then convert the section title to a one-hot vector. Then we train LR, XGBoost, and Adaboost on the position features.

Next, we focus on the aspect of text content since the texts of CT and RT indicate the content in detail. First, the texts of CT and RT are preprocessed, such as extracting text from XML file, stop word removal, and word tokenization. Then they are represented by word embeddings and mapped to a dense vector space by FastText. The architecture of FastText is shown in Figure 3 where $x_1, x_2, ..., x_{N-1}, x_N$ represent the n-gram features and each feature is the average of word embeddings. The hidden layer is obtained from the average of $x_1, x_2, ..., x_{N-1}, x_N$. Then the output layer is fully connected to the hidden layer and finally obtain the predicted label by the hierarchical softmax. The reason that we choose FastText as our classifier based on content features is that FastText is relatively lighter than other text classifiers and can avoid overfitting since the training set is small.

3.1.3 Task2

Task2 is a summarization task, and we apply two extractive methods in this paper.

Extractive summarization based on DPPs: This method assumes that each document is a set of sentences, and the process of extracting the summary is to extract the highest quality subset from the set of sentences. To achieve this extraction process, we first represent the document as a matrix L representing the relationship between sentences and then apply the DPPs sampling algorithm to extract candidate sentences. The matrix L is constructed by the Quality-Diversity (QD) model and Sent2Vec (SV) model.

In the Quality-Diversity model, matrix L can be calculated by:

$$L_{ij} = q_i Sim_{ij} q_j$$

where q_i is the quality of each sentence which can be calculated by the features we selected, such as Sentence Length (SL), Sentence Position (SP) and Sentence Coverage (SC). Sim_{ij} represents the similarity between sentences, which can be imple-

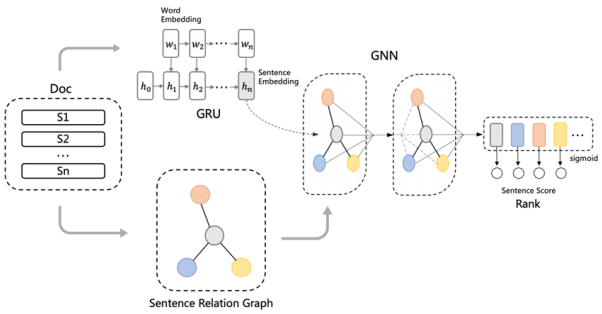

Figure 4: Extractive summarization based on GCN in Task2.

mented as

$$Sim_{ij} = \varphi_i^T \varphi_j \in [0, 1]$$

where φ_i is the diversity vector of a single sentence.

In the Sent2Vec model, we construct matrix L by

$$L_{ij} = B_i^T B_j$$

where B is the sentence vector obtained from the Sent2Vec model.

By constructing matrix L, we can apply the DPPs sampling algorithm to select sentences, the extracted summaries have both high-quality and low-similarity. The details of DPPs can be referred to the work of Kulesza and Taskar (2012).

Extractive summarization based on GNN: We propose an extractive summarization method based on GCN and GAT (Figure 5). As shown in Figure 4, we first build a sentence relation graph based on sentence similarity, calculated by cosine similarity. The similarity graph can objectively reflect the association between sentences, including keywords and sentence similarity information. The graphs and low-level sentence representations compressed by GRU are fed into GCN and GAT. Each node in the undirected graph is a sentence, which is connected to another sentence if their similarity is greater than 0.2, and the origin node feature is the last hidden layer of GRU. Graph convolution can leverage the feature information of the node itself and the structure information of the graph. In the L-layer convolution network, $H^{(l)}$ represents the hidden features of the l^{th} layer, parameterized by a weight matrix $W^{(l)}$. And $\tilde{A}$ is symmetrically normalized from the graph adjacency matrix A. After a non-linear function(ReLU), we obtain advanced representations as the final scoring features.

$$f(H^{(l)}, A) = \sigma(\tilde{A}H^{(l)}W^{(l)})$$

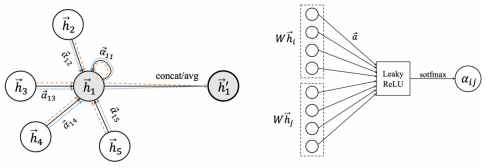

Figure 5: **Left:** Multi-head attention (with 3 heads) computations apply on node 1 and its neighborhood. $\vec{h}_1'$ is obtained by concatenating or averaging from the aggregated features of each head. **Right:** The attention mechanism $a(W\vec{h}_i, W\vec{h}_j)$, and activated by LeakyReLU.

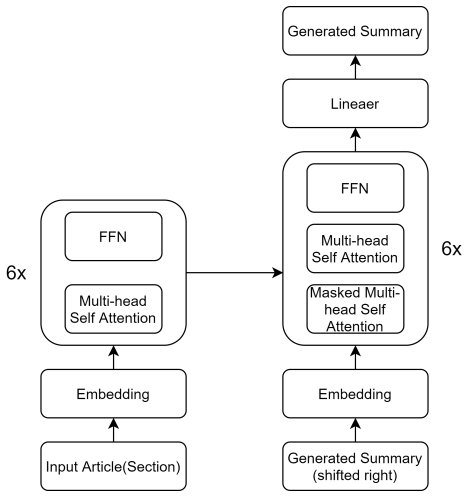

Figure 6: T5 is actually a transformer pretrained on the large corpus. We fine-tuned it for abstractive summarization task.

In the training period, we select the sentences most similar to the community summary from the RP as the summary sentences. The selected sentences are labeled as 1, while the rest sentences are labeled as 0. Then the model is trained as a binary classifier. Finally, we greedily select the highest-scoring sentences from the sentence set.

3.2 LongSumm

For the LongSumm shared task, we use three methods based on our forementioned summarization methods for Task2 of CL-SciSumm in this paper.

3.2.1 T5 Fine-tuning

Although we have divided the dataset into section-wise samples and obtain more than 30000 section-summary pairs, it is still not sufficient to train an abstractive model from scratch. Therefore we use the pre-training and fine-tuning method to deal with this problem. As shown in Figure 6, T5 (Raffel et al., 2019b) is a transformer-liked pre-trained model that has great performance when transfer

to a summarization task. It treats every NLP task as a text-to-text task and does both unsupervised pre-training and supervised multi-task pre-training on the large corpus.

3.2.2 DPPs Sampling

This method is based on DPPs sampling, which is similar to the method in Task2 of the CL-SciSumm shared task. We utilize two models to construct matrix L, that is, the Quality-Diversity (QD) model and the Sent2Vec (SV) model. Then DPPs sampling can automatically select the candidate sentences with high quality.

3.2.3 GRU-GCN/GAT

This method contains two parts: an RNN model and a GCN/GAT model. When processing the original text data, we use GRU to compress the sequences. And a similarity graph is constructed for each sentence group as described in 3.1.3, together with the sentence representation as sentence node feature are fed into GCN or GAT. Then we apply the method of supervised training as a reference to the binary classification and select the highest-scored sentences according to the training results.

4 Experiments and Results

4.1 CL-SciSumm

4.1.1 Task1A

In our previous work (Li et al., 2019), we have extracted many kinds of features through various methods. In terms of semantic information, the features are word vector, word-cos, and Lin and Jcn in WordNet. Some traditional features are also used such as IDF and Jaccard similarity, considering that with the increase of the number of topics in the LDA model, the topic vector will gradually become sparse. This time, we abandon LDA and LDA-cos features and introduce the LDA-Jaccard similarity, which can improve the discrimination performance of LDA when the topic vector is sparse and focus on the similarity in the same topic. Based on the original fusion method, there are four new fusion methods by increasing the training set and adjusting the hyper-parameters, which are, Voting-1.2, Voting-2.1, Jaccard-Focused, and Jaccard-Cascade.

LDA model topic size is set to 600, and the pre-training word vector size is set to 300. In the case of high-dimensional LDA, although the word distribution in the topic becomes very sparse, the performance has been improved. Table 4 shows the parameter settings of the four multi-feature fusion methods.

Method	Precision	Recall	F1 Score
V1.2	0.0693	**0.2658**	0.1100
V2.1	0.0604	0.2308	0.0958
JF	**0.0698**	0.2650	**0.1105**
JC	0.0605	0.2331	0.0960

Table 1: Task1A experiment results. V1.2, V2.1, JF, JC are Voting-1.2, Voting-2.1, Jaccard-Focused, Jaccard-Cascade respectively.

Facet	Proportion
Aim Citation	0.082
Method Citation	0.718
Hypothesis Citation	0.024
Result Citation	0.138
Implication Citation	0.080
Multi-facet	0.074

Table 2: Facet distribution of the training set in Task1B.

As shown in Table 1, the performance of Jaccard-Focused is the best among the four methods. At the same time, there is a big gap between the precision and the recall rate. It is because we manually specify that top-N sentences are answers, so the program finds more sentences in general, so the recall rate is higher than the precision rate.

4.1.2 Task1B

For XGBoost (POS-XGB), we set the learning rate to 0.3, max depth to 1; for Adaboost (POS-ADB), we use the decision tree as the weak learner with max depth 2, learning rate 0.3; for LR (POS-LR), we set the learning rate to 0.3. We also implement a voting method (POS-Vote) based on these base classifiers. As for FastText (CON-CT-FastText and CON-RT-FastText) applied on content features, the CT and RT length are 40 and 50 respectively. The size of word embedding, hidden layer and output layer are 128, 256 and 2 respectively. We use Adam as the optimizer with learning rate 0.0001, and train for 50 iterations. Finally, we combine the classifiers on position features and content features via a voting method (CON-POS-Vote). Both the vote methods mentioned above obey the majority rule.

Since Task1B is a multi-label classification task and the training set is severely imbalanced, as shown in Table 2, we randomly sample an equal number of negative samples for each discourse facet, then train five independent classifiers, respectively. When predicting the test set, we select

Method	Precision	Recall	F1 Score
POS-ADB	0.3088	0.1439	0.1963
POS-LR	0.3685	0.1813	0.2430
POS-Vote	0.4464	0.1831	0.2597
POS-XGB	0.4660	**0.2331**	**0.3108**
CON-CT-FastText	0.4994	0.2047	0.2904
CON-RT-FastText	0.4624	0.1990	0.2783
CON-POS-Vote	**0.5533**	0.1917	0.2847

Table 3: Task1B experiment results.

at most top 2 facets with the highest probability.

Table 3 shows the results of Task1B. We find that CON-POS-Vote has the best Precision, while POS-XGB performs best on Recall and F1 Score. The performance of FastText based on content features is better than most of machine learning methods based on position features. And CTs contain more information indicating the facet than RT.

4.1.3 Task2

In DPPs sampling, Sentence Length (SL), Sentence Position (SP), and Sentence Coverage (SC) are selected as features to calculate the quality of sentences, and the summary compression ratio is set to 20%. For the GCN method, we pick the top 50k words sorted by the frequency from the vocabulary of the original text. We select a sentence subset with the largest ROUGE score as the target for extractive summarization. Based on the greedy algorithm, the sentence with the largest ROUGE score is taken out one by one as a positive sample and added to the extractive summary set until the set cannot increase the score. After cleaning the RP, we rank the sentences by the output score, and then the summaries are generated. Table 5 shows the result on the test set.

From Table5 we can see that GCN based methods perform better than DPPs on various metrics of three different gold summaries. It indicates that end to end supervised learning method can extract better feature than human, even the supervised signals are constructed indirectly (we construct extractive summarization training data from human-write summarization dataset). Although DPPs performs well on improving the diversity of summaries, its ability to evaluate the quality of sentence comes from handcrafted feature, which generalize worse.

4.2 LongSumm

4.2.1 Data preprocessing

The training data set is composed of abstractive parts and extractive parts. The abstractive summarization data are from published papers and blogs which contain around 700 articles with an average of 31.7 sentences per summary and an average of 21.6 words per sentence. The extractive data are from Lev et al. (2019) which have 1705 paper-summary pairs. For each paper, it provides a summary with 30 sentences and 990 words on average. The LongSumm shared task is characterized by long input and output with a high compression ratio. So we choose a mix-and-divide method to deal with it:

1. To make full use of all data samples, we mix abstractive and extractive data.

2. Transform the full paper level summarization into short document summarization by dividing all article-summary pairs into section-summary pairs.

3. Relabel all samples for abstractive models and extractive models.

The first step is easy to understand. The second step is achieved as follows: with PDF parser, we can identify sections in the paper; the highest Jaccard similarity among all pairs between sections sentences and summary sentences is used as section-sentence Jaccard similarity; each summary sentence is allocated to the section which has the highest section-sentence Jaccard similarity with it. Other co-occurrence based metrics like ROUGE (Gidiotis and Tsoumakas, 2020) or BLEU can also be applied but we choose jaccard because of its simplicity(these metrics usually lead to the same allocation). We get 30230 section-summary pairs in total. At last, we build two datasets with different types:

1. For extractive models, sentences in a section that have the highest Jaccard similarity with summary sentences are labeled to be extracted.

2. For abstractive models, there is no need to process abstractive samples. Extractive samples are processed according to Zhang et al. (2019). For the long section, we use textrank to extract some sentences as a summary and exclude these sentences from the section. This preprocessing trick can prevent the abstractive model from learning to copy input. For a short section, we do not exclude summary sentences from the section.

Feature	V1.2		V2.1		JF		JC	
	w	p	w	p	w	p	w	p
Idf similarity	1	12	0.5	5	0.6	16	0.5	16
Idf context similarity			0.8	3	0.5	15	0.4	10
Jaccard similarity	1	5	0.5	6	JS	7		
Jaccard context similarity			0.5	8	0.7	16	0.6	16
Word vector	1	8	0.5	7	0.5	26		
word-cos	1	10	0.7	7	0.5	26	0.5	10
LDA-Jaccard	1	12	0.4	7				
lin			0.5	5				
jcn					0.6	11		

Table 4: Parameters in multi-feature fusion methods in Task1A. V1.2, V2.1, JF, JC are Voting-1.2, Voting-2.1, Jaccard-Focused, Jaccard-Cascade respectively

method	**abstract**		**community**		**human**	
	R2	**RSU4**	**R2**	**RSU4**	**R2**	**RSU4**
Jaccard-Cascade_GCN	0.19648	**0.10392**	0.2195	0.14174	0.19117	0.13984
Jaccard-Cascade_QD-DPPs	0.14483	0.09439	0.1492	0.09525	0.13961	0.11623
Jaccard-Cascade_SV-DPPs	0.0981	0.06849	0.17051	0.10209	0.11548	0.09381
Jaccard-Focused_GCN	0.19931	0.09956	**0.24549**	**0.15071**	**0.2042**	**0.14162**
Jaccard-Focused_QD-DPPs	0.12206	0.08266	0.16443	0.09663	0.12376	0.09957
Jaccard-Focused_SV-DPPs	0.12196	0.07936	0.16491	0.09954	0.15772	0.11616
Voting-1.1_GCN	**0.20643**	0.09324	0.24119	0.14578	0.17673	0.11583
Voting-1.1_QD-DPPs	0.1345	0.07303	0.14744	0.09072	0.10559	0.08623
Voting-1.1_SV-DPPs	0.12132	0.06753	0.18003	0.09822	0.13236	0.0937
Voting-2.0_GCN	0.18042	0.07915	0.23088	0.13093	0.17653	0.10737
Voting-2.0_QD-DPPs	0.08908	0.05948	0.17384	0.09739	0.09949	0.07156
Voting-2.0_SV-DPPs	0.14098	0.08039	0.15819	0.09075	0.11406	0.07844

Table 5: Task2 experiment results. JC means Jaccard-Cascade. JF stands for Jaccard-Focused. V1.2 and V2.1 are Voting-1.2 and Voting-2.1 respectively.

We divide the dataset into train/dev/test for comparing different models in this report. ROUGE evaluation is given on the divided test set and we use all 30230 samples for training when inferring on the blind test set.

4.2.2 Result

The result of the LongSumm shared task is illustrated in Table 6.

For model T5, we use the small version which has about 60 million parameters. All input sections are truncated to a maximum of 1024 words. The model is fine-tuned for 5 epochs on the section-wise dataset with a learning rate of 1e-4. The batch size is 32 and we use gradient accumulation to achieve it on a single GPU. Then, we attempt different ways to process the original data, expecting to find the proper input for the model.

1. Construct summary, as mentioned above, all data is transferred to abstractive data.

2. Original summary, as the name suggests, original data are used as input. Because many sections do not have corresponding summaries, there are fewer samples can be utilized, but some corresponding summaries are relatively longer.

3. Original+Construct summary, this method merges the original section and the construct section.

In order to generate the summary as long as possible within the limitation of summary length, we design two plans to process the generated summaries. Plan A simply merges the first sentence of summaries that are generated from different sections. Plan B extracts at most three sentences from each summary, for those with fewer words, we can use all sentences. Also, the merged summary is truncated to 600 words if the word count exceeds the limit.

As for DPPs, because the LongSumm task focuses on a long summary, we change the document compression ratio to control the summary length, we set the ratio to 20% and 30%. For the QD method, we select Sentence Length (SL), Sentence Position (SP), and Sentence Coverage (SC) as features and merge them, which can calculate sentence quality.

As for GRU-GCN/GAT, we divide each paper

methods	submission	Description	R1 f	R1 r	R2 f	R2 r	RL f	RL r
Pretrained Language Model Based	0	Construct-A	0.424	0.373	0.120	0.104	0.172	0.149
	1	Original-A	0.487	0.490	0.134	0.136	0.182	0.184
	2	Construct+Original-A	0.417	0.364	0.108	0.094	0.171	0.148
	3	Construct-B	0.489	0.490	0.135	0.136	0.183	0.184
	4	Original-B	0.488	0.494	0.134	0.137	0.183	0.185
	5	Construct+Original-B	0.487	0.487	0.135	0.136	0.183	0.183
GNN Based	6	GRU+GAT-sim	0.479	0.491	0.143	0.146	0.182	0.186
	7	GRU+GCN-sim	**0.490**	**0.497**	**0.151**	**0.152**	**0.201**	**0.204**
DPPs Based	8	QD-DPPs-20	0.448	0.428	0.102	0.097	0.169	0.161
	9	QD-DPPs-30	0.435	0.415	0.103	0.099	0.162	0.154
	10	SV-DPPs-20	0.452	0.427	0.109	0.103	0.165	0.156
	11	SV-DPPs-30	0.451	0.428	0.120	0.114	0.170	0.161

Table 6: LongSumm test set results. ROUGE_f and ROUGE_r are f1 value and recall of ROUGE results.

into sections, since sections are the natural division of paper, and match each section to its gold summaries. For every section, its relation graph is constructed and system summaries are extracted by sentence scores. After we get the section summaries, paper summaries are concatenated by ranking sentences from sections. In our work, GAT has more parameters, thus are more difficult to converge, and the advantage to learn graph structure is weakened since section graphs are rather small, which explains why attention mechanism does not do better than GCN in some way.

The results on the test set show that extractive summarization model using the GCN method performs the best on long summary task and the performance of T5 and DPPs is slightly worse than GCN. Generally speaking, the ROUGE value of abstractive summaries is lower than that of extractive summaries. But as an abstractive summarization model, T5 can compress more semantic information to generate the summary closer to an artificial summary. As for DPPs, as an unsupervised model, it uses hand-constructed features to rank sentences. The sentence quality obtained by this is not accurate. GNN uses RNN to model sentences, and considers sentence diversity in the learning process of neural network. So the ability to measure sentence quality is weaker than GNN. However, DPPs is able to work well under the situation where the training data is lacked.

5 Conclusion and Future Work

In the CL-SciSumm shared task, Jaccard-Focused performs better than other methods in Task1A. In future work, we will try to use the knowledge graph and GNN for better expression of semantic and structure information. In Task1B, POS-XGB performs the best, which shows that the position features contributes more than the content features. In the future, more information can be extracted and fused to obtain richer features, or combined with some hand-craft rules to assist the classification. In Task 2, GCN shows great potential to perform the summarization task. We expect the neural network language models to make contributions to obtain more meaningful semantic representation for sentences against statistical features. In the LongSumm shared task, model T5 and extractive summarization model based on GCN perform well on the official data set, and DPPs still has great potential, we expect to provide more features or modify the sampling processes so as to improve the performance of our models. What's more, in this paper we mainly focus on how to extract/generate section-wise summaries with high quality and diversity, but how to pick and combine these summaries is also an interesting work to be done.

References

M. K. Chandrasekaran, G. Feigenblat, Hovy. E., A. Ravichander, M. Shmueli-Scheuer, and A De Waard. forthcoming. 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)*.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3844–3852. Curran Associates, Inc.

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*.

Cicero Dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78.

Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

Alexios Gidiotis and Grigorios Tsoumakas. 2020. A divide-and-conquer approach to the summarization of academic articles.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Alex Kulesza and Ben Taskar. 2012. Determinantal point processes for machine learning. *arXiv preprint arXiv:1207.6083*.

Matt Kusner, Y. Sun, N.I. Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, pages 957–966.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

Guy Lev, Michal Shmueli-Scheuer, Jonathan Herzig, Achiya Jerbi, and David Konopnicki. 2019. TalkSumm: A dataset and scalable annotation method for scientific paper summarization based on conference talks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2125–2131, Florence, Italy. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zet-tlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural lan-guage generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Lei Li, Junqi Chi, Moye Chen, Zuying Huang, Yingqi Zhu, and Xiangling Fu. 2018. Cist@ clscisumm-18: Methods for computational linguistics scientific citation linkage, facet classification and summarization. In *BIRNDL@ SIGIR*.

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 *BIRNDL@ SIGIR*, pages 196–207.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *AAAI*, volume 16, pages 2793–2799.

Hyeoun-Ae Park. 2013. An introduction to logistic regression: From basic concepts to interpretation with particular attention to nursing domain. *Journal of Korean Academy of Nursing*, 43:154–164.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019a. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019b. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*.

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, Lisbon, Portugal. Association for Computational Linguistics.

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

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 452–462, Vancouver, Canada. Association for Computational Linguistics.

Li Yujian and Liu Bo. 2007. A normalized levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1091–1095.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2019. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Ming Zhong, Pengfei Liu, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2019. Searching for effective neural extractive summarization: What works and what's next. *arXiv preprint arXiv:1907.03491*.