TWAG: A Topic-guided Wikipedia Abstract Generator

Fangwei Zhu[1,2], **Shangqing Tu**[3], **Jiaxin Shi**[1,2], **Juanzi Li**[1,2], **Lei Hou**[1,2*] **and Tong Cui**[4]
[1]Dept. of Computer Sci.&Tech., BNRist, Tsinghua University, Beijing 100084, China
[2]KIRC, Institute for Artificial Intelligence, Tsinghua University
[3]School of Computer Science and Engineering, Beihang University
[4]Noah's Ark Lab, Huawei Inc.
{zfw19@mails.,shijx16@mails,lijuanzi@,houlei@}tsinghua.edu.cn
tsq@buaa.edu.cn,cuitong5@huawei.com

Abstract

Wikipedia abstract generation aims to distill a Wikipedia abstract from web sources and has met significant success by adopting multi-document summarization techniques. However, previous works generally view the abstract as plain text, ignoring the fact that it is a description of a certain entity and can be decomposed into different topics. In this paper, we propose a two-stage model TWAG that guides the abstract generation with topical information. First, we detect the topic of each input paragraph with a classifier trained on existing Wikipedia articles to divide input documents into different topics. Then, we predict the topic distribution of each abstract sentence, and decode the sentence from topic-aware representations with a Pointer-Generator network. We evaluate our model on the WikiCatSum dataset, and the results show that TWAG outperforms various existing baselines and is capable of generating comprehensive abstracts. Our code and dataset can be accessed at https://github.com/THU-KEG/TWAG

1 Introduction

Wikipedia, one of the most popular crowd-sourced online knowledge bases, has been widely used as the valuable resources in natural language processing tasks such as knowledge acquisition (Lehmann et al., 2015) and question answering (Hewlett et al., 2016; Rajpurkar et al., 2016) due to its high quality and wide coverage. Within a Wikipedia article, its abstract is the overview of the whole content, and thus becomes the most frequently used part in various tasks. However, the abstract is often contributed by experts, which is labor-intensive and prone to be incomplete.

In this paper, we aim to automatically generate Wikipedia abstracts based on the related documents collected from referred websites or search engines, which is essentially a multi-document summarization problem. This problem is studied in both extractive and abstractive manners.

The extractive models attempt to select relevant textual units from input documents and combine them into a summary. Graph-based representations are widely exploited to capture the most salient textual units and enhance the quality of the final summary (Erkan and Radev, 2004; Mihalcea and Tarau, 2004; Wan, 2008). Recently, there also emerge neural extractive models (Yasunaga et al., 2017; Yin et al., 2019) utilizing the graph convolutional network (Kipf and Welling, 2017) to better capture inter-document relations. However, these models are not suitable for Wikipedia abstract generation. The reason is that the input documents collected from various sources are often noisy and lack intrinsic relations (Sauper and Barzilay, 2009), which makes the relation graph hard to build.

The abstractive models aim to distill an informative and coherent summary via sentence-fusion and paraphrasing (Filippova and Strube, 2008; Banerjee et al., 2015; Bing et al., 2015), but achieve little success due to the limited scale of datasets. Liu et al. (2018) proposes an extractive-then-abstractive model and contributes WikiSum, a large-scale dataset for Wikipedia abstract generation, inspiring a branch of further studies (Perez-Beltrachini et al., 2019; Liu and Lapata, 2019; Li et al., 2020).

The above models generally view the abstract as plain text, ignoring the fact that Wikipedia abstracts describe certain entities, and the structure of Wikipedia articles could help generate comprehensive abstracts. We observe that humans tend to describe entities in a certain domain from several topics when writing Wikipedia abstracts. As illustrated in Figure 1, the abstract of the *Arctic Fox* contains its adaption, biology taxonomy and geographical distribution, which is consistent with

[*] Corresponding Author

Abstract	Content Table
The Arctic fox (Vulpes lagopus), also known as the white fox, polar fox, or snow fox, is a small fox native to the Arctic regions of the Northern Hemisphere and common throughout the Arctic tundra biome. It is well adapted to living in cold environments, and is best known for its thick, warm fur that is also used as camouflage. It has a large and very fluffy tail. In the wild, most individuals do not live past their first year but some exceptional ones survive up to 11 years. Its body length ranges from 46 to 68 cm (18 to 27 in), with a generally rounded body shape to minimize the escape of body heat.	2 Adaptations 2.1 Sensory modalities 2.2 Physiology 3 Size 4 Taxonomy 4.1 Origins 4.2 Subspecies 5 Distribution and habitat 5.1 Migrations and travel

Figure 1: An example of Wikipedia article *Arctic Fox*. The abstract contains three orthogonal topics about an animal: Description, Taxonomy and Distribution. The right half is part of the article's content table, showing section labels related to different topics.

the content table. Therefore, given an entity in a specific domain, generating abstracts from corresponding topics would reduce redundancy and produce a more complete summary.

In this paper, we try to utilize the topical information of entities within its domain (Wikipedia categories) to improve the quality of the generated abstract. We propose a novel two-stage Topic-guided Wikipedia Abstract Generation model (**TWAG**). TWAG first divides input documents by paragraph and assigns a topic for each paragraph with a classifier-based topic detector. Then, it generates the abstract in a sentence-wise manner, i.e., predicts the topic distribution of each abstract sentence to determine its topic-aware representation, and decodes the sentence with a Pointer-Generator network (See et al., 2017).

We evaluate TWAG on the **WikiCatSum** (Perez-Beltrachini et al., 2019) dataset, a subset of the **WikiSum** containing three distinct domains. Experimental results show that it significantly improves the quality of abstract compared with several strong baselines.

In conclusion, the contributions of our work are as follows:

- We propose TWAG, a two-stage neural abstractive Wikipedia abstract generation model utilizing the topic information in Wikipedia, which is capable of generating comprehensive abstracts.
- We simulate the way humans recognize entities, using a classifier to divide input documents into topics, and then perform topic-aware abstract generation upon the predicted topic distribution of each abstract sentence.
- Our experiment results against 4 distinct baselines prove the effectiveness of TWAG.

2 Related Work

2.1 Multi-document Summarization

Multi-document summarization is a classic and challenging problem in natural language processing, which aims to distill an informative and coherent summary from a set of input documents. Compared with single-document summarization, the input documents may contain redundant or even contradictory information (Radev, 2000).

Early high-quality multi-document summarization datasets are annotated by humans, e.g., datasets for Document Understanding Conference (DUC) and Text Analysis Conference (TAC). These datasets are too small to build neural models, and most of the early works take an extractive method, attempting to build graphs with inter-paragraph relations and choose the most salient textual units. The graph could be built with various information, e.g., TF-IDF similarity (Erkan and Radev, 2004), discourse relation (Mihalcea and Tarau, 2004), document-sentence two-layer relations (Wan, 2008), multi-modal (Wan and Xiao, 2009) and query information (Cai and Li, 2012). Recently, there emerge attempts to incorporate neural models, e.g., Yasunaga et al. (2017) builds a discourse graph and represents textual units upon the graph convolutional network (GCN) (Kipf and Welling, 2017), and Yin et al. (2019) adopts the entity linking technique to capture global dependencies between sentences and ranks the sentences with a neural graph-based model.

In contrast, early abstractive models using sentence-fusion and paraphrasing (Filippova and Strube, 2008; Banerjee et al., 2015; Bing et al., 2015) achieve less success. Inspired by the recent success of single-document abstractive models (See et al., 2017; Paulus et al., 2018; Gehrmann et al., 2018; Huang et al., 2020), some works (Liu et al., 2018; Zhang et al., 2018) try to transfer single-document models to multi-document settings to alleviate the limitations of small-scale datasets. Specifically, Liu et al. (2018) defines Wikipedia generation problem and contributes the large-scale WikiSum dataset. Fabbri et al. (2019) constructs a middle-scale dataset named Multi-

News and proposes an extractive-then-abstractive model by appending a sequence-to-sequence model after the extractive step. Li et al. (2020) models inter-document relations with explicit graph representations, and incorporates pre-trained language models to better handle long input documents.

2.2 Wikipedia-related Text Generation

Sauper and Barzilay (2009) is the first work focusing on Wikipedia generation, which uses Integer Linear Programming (ILP) to select the useful sentences for Wikipedia abstracts. Banerjee and Mitra (2016) further evaluates the coherence of selected sentences to improve the linguistic quality.

Liu et al. (2018) proposes a two-stage extractive-then-abstractive model, which first picks paragraphs according to TF-IDF weights from web sources, then generates the summary with a transformer model by viewing the input as a long flat sequence. Inspired by this work, Perez-Beltrachini et al. (2019) uses a convolutional encoder and a hierarchical decoder, and utilizes the Latent Dirichlet Allocation model (LDA) to render the decoder topic-aware. HierSumm (Liu and Lapata, 2019) adopts a learning-based model for the extractive stage, and computes the attention between paragraphs to model the dependencies across multiple paragraphs. However, these works view Wikipedia abstracts as plain text and do not explore the underlying topical information in Wikipedia articles.

There are also works that focus on generating other aspects of Wikipedia text. Biadsy et al. (2008) utilizes the key-value pairs in Wikipedia infoboxes to generate high-quality biographies. Hayashi et al. (2021) investigates the structure of Wikipedia and builds an aspect-based summarization dataset by manually labeling aspects and identifying the aspect of input paragraphs with a fine-tuned RoBERTa model (Liu et al., 2019). Our model also utilizes the structure of Wikipedia, but we generate the compact abstract rather than individual aspects, which requires the fusion of aspects and poses a greater challenge to understand the connection and difference among topics.

3 Problem Definition

Definition 1 ***Wikipedia abstract generation*** *accepts a set of paragraphs*[1] $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ *of size n as input, and outputs a Wikipedia abstract $\mathcal{S} = (s_1, s_2, \ldots, s_m)$ with m sentences. The goal is to find an optimal abstract $\mathcal{S}^*$ that best concludes the input, i.e.,*

$$\mathcal{S}^* = \arg\max_{\mathcal{S}} P(\mathcal{S}|\mathcal{D}) \tag{1}$$

Previous works generally view $\mathcal{S}$ as plain text, ignoring the semantics in Wikipedia articles. Before introducing our idea, let's review how Wikipedia organizes articles.

Wikipedia employs a hierarchical open category system to organize millions of articles, and we name the top-level category as domain. As for a Wikipedia article, we concern three parts, i.e., the abstract, the content table, and textual contents. Note that the content table is composed of several section labels $\{l\}$, pairing with corresponding textual contents $\{p\}$. As illustrated in Figure 1, the content table indicates different aspects (we call them topics) of the article, and the abstract semantically corresponds to these topics, telling us that topics could benefit the abstract generation.

However, general domains like *Person* or *Animal* consist millions of articles with diverse content tables, making it not feasible to simply treat section labels as topics. Considering that articles in specific domains often share several salient topics, we manually merge similar section labels to convert the sections titles to a set of topics. Formally, the topic set is denoted as $\mathcal{T} = \{T_1, T_2, ..., T_{n_t}\}$ of size n_t, where each topic $T_i = \{l_i^1, l_i^2, \ldots, l_i^m\}$.

Now, our task can be expressed with a topical objective, i.e.,

Definition 2 *Given the input paragraphs $\mathcal{D}$, we introduce the latent topics $\mathcal{Z} = \{z_1, z_2, \ldots, z_n\}$, where $z_i \in \mathcal{T}$ is the topic of i-th input paragraph d_i, and our objective of Wikipedia abstract generation is re-written as*

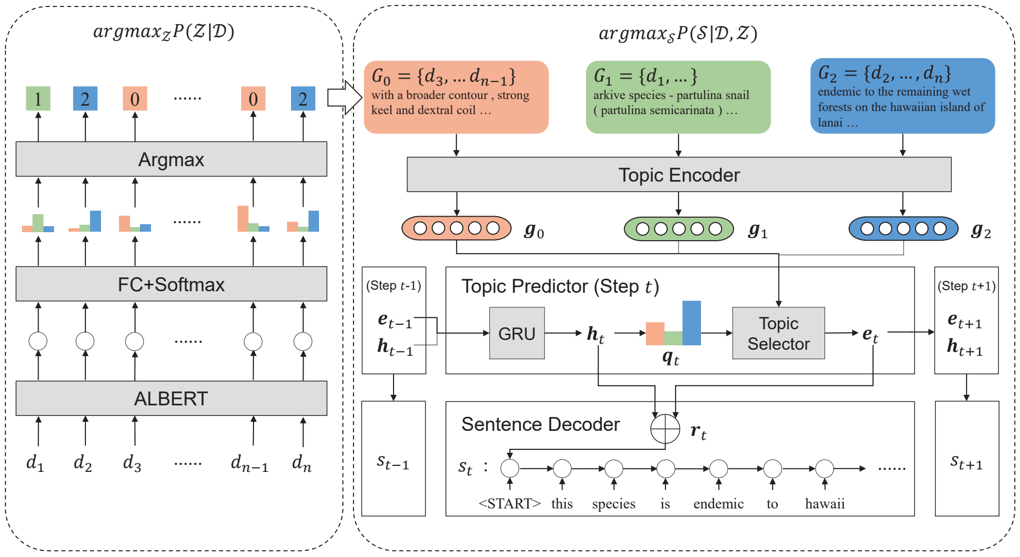
$$\mathcal{S}^* = \arg\max_{\mathcal{Z}} P(\mathcal{Z}|\mathcal{D}) \arg\max_{\mathcal{S}} P(\mathcal{S}|\mathcal{D}, \mathcal{Z}). \tag{2}$$

Therefore, the abstract generation could be completed with two sub-tasks, i.e., topic detection to optimize $\arg\max_{\mathcal{Z}} P(\mathcal{Z}|\mathcal{D})$ and topic-aware abstract generation to optimize $\arg\max_{\mathcal{S}} P(\mathcal{S}|\mathcal{D}, \mathcal{Z})$.

4 The Proposed Method

As shown in Figure 2, our proposed TWAG adopts a two-stage structure. First, we train a topic detector based on existing Wikipedia articles to predict the topic of input paragraphs. Second, we group the

[1]The input documents can be represented by textual units with different granularity, and we choose paragraph as it normally expresses relatively complete and compact semantics.



$argmax_{\mathcal{Z}} P(\mathcal{Z}|\mathcal{D})$
1
2
0
......
0
2
Argmax
......
FC+Softmax
......
ALBERT
d_1
d_2
d_3
......
d_{n-1}
d_n
$argmax_{\mathcal{S}} P(\mathcal{S}|\mathcal{D},\mathcal{Z})$
$G_0 = \{d_3, \dots d_{n-1}\}$
with a broader contour , strong keel and dextral coil …
$G_1 = \{d_1, \dots\}$
arkive species - partulina snail (partulina semicarinata) …
$G_2 = \{d_2, \dots, d_n\}$
endemic to the remaining wet forests on the hawaiian island of lanai …
Topic Encoder
g_0
g_1
g_2
(Step t-1)
e_{t-1}
h_{t-1}
Topic Predictor (Step t)
GRU
h_t
q_t
Topic Selector
e_t
(Step t+1)
e_{t+1}
h_{t+1}
s_{t-1}
Sentence Decoder
r_t
s_t :
......
<START>
this
species
is
endemic
to
hawaii
s_{t+1}

Figure 2: The TWAG framework. We use an example domain with 3 topics for illustration. The left half is the topic detector which attempts to find a topic for each input paragraph, and the right half is the topic-aware abstract generator to generate the abstract by sentence based on input paragraphs and their predicted topics.

input paragraphs by detected topics to encode them separately, and generate the abstract in a sentence-wise manner. In each step, we predict the topic distribution of the current sentence, fuse it with the global hidden state to get the topic-aware representation, and generate the sentence with a copy-based decoder. Next, we will detail each module.

4.1 Topic Detection

The topic detector aims to annotate input paragraphs with their optimal corresponding topics. To formalize, given the input paragraphs $\mathcal{D}$, Det returns its corresponding topics $\mathcal{Z} = \{z_1, z_2, \dots, z_n\}$, i.e.,

$$\mathcal{Z} = \text{Det}(\mathcal{D}) \tag{3}$$

We view topic detection as a classification problem. For each paragraph $d \in \mathcal{D}$, we encode it with ALBERT(Lan et al., 2019) and then predict its topic z with a fully-connected layer, i.e.,

$$\mathbf{d} = \text{ALBERT}(d) \tag{4}$$

$$z = \arg\max(\text{linear}(\mathbf{d})) \tag{5}$$

where $\mathbf{d}$ is the vector representation of d, and we fine-tuned the ALBERT model on a pretrained version.

4.2 Topic-aware Abstract Generation

Topic-aware abstract generator utilizes the input paragraphs $\mathcal{D}$ and the detected topics $\mathcal{Z}$ to generate the abstract. Specifically, it contains three modules: a topic encoder to encode the input paragraphs into topical representations, a topic predictor to predict the topic distribution of abstract sentences and generate the topic-aware sentence representation, and a sentence decoder to generate abstract sentences based on the topic-aware representations.

4.2.1 Topic Encoder

Given the input paragraphs $\mathcal{D}$ and the detected topics $\mathcal{Z}$, we concatenate all paragraphs belonging to the same topic T_k to form a topic-specific text group (TTG) $\mathcal{G}_k$, which contains salient information about a certain topic of an entity:

$$\mathcal{G}_k = \text{concat}(\{d_i | z_i = T_k\}). \tag{6}$$

To further capture hidden semantics, we use a bidirectional GRU to encode the TTGs:

$$\mathbf{g}_k, \mathbf{U}_k = \text{BiGRU}(\mathcal{G}_k). \tag{7}$$

$\mathbf{g}_k$ is the final hidden state of the $\mathcal{G}_k$, and $\mathbf{U}_k = (\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{n_{G_k}})$ represents the hidden state of each token in $\mathcal{G}_k$, where n_{G_k} denotes the number of tokens in $\mathcal{G}_k$.

4.2.2 Topic Predictor

After encoding the topics into hidden states, TWAG tackles the decoding process in a sentence-wise manner:

$$\arg\max_{\mathcal{S}} P(\mathcal{S}|\mathcal{D},\mathcal{Z}) = \prod_{i=1}^{m} \arg\max_{s_i} P(s_i|\mathcal{D},\mathcal{Z},s_{<i}) \tag{8}$$

To generate the abstract $\mathcal{S}$, we first predict the topic distribution of every sentence s_i with a GRU decoder. At each time step t, the topic predictor produces a global hidden state $\mathbf{h}_t$, and then estimates the probability distribution $\mathbf{q}_t$ over topics.

$$\mathbf{h}_t = \text{GRU}(\mathbf{h}_{t-1}, \mathbf{e}_{t-1}) \quad (9)$$
$$\mathbf{q}_t = \text{softmax}(\text{linear}(\mathbf{h}_t)) \quad (10)$$

where $\mathbf{e}_{t-1}$ denotes the topical information in the last step. $\mathbf{e}_0$ is initialized as an all-zero vector, and $\mathbf{e}_t$ could be derived from $\mathbf{q}_t$ in two ways.

The first way named **hard topic**, is to directly select the topic with the highest probability, and take its corresponding representation, i.e.,

$$\mathbf{e}_t^{hard} = \mathbf{g}_{\arg\max_i(q_i)}. \quad (11)$$

The second way named **soft topic**, is to view every sentence as a mixture of different topics, and take the weighted sum over topic representations, i.e.,

$$\mathbf{e}_t^{soft} = \mathbf{q}_t \cdot \mathbf{G} \quad (12)$$

where $\mathbf{G} = (\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_{n_t})$ is the matrix of topic representations. With the observation that Wikipedia abstract sentences normally contain mixed topics, we choose the soft topic mechanism for our model (see Section 5.3 for details).

Finally, we compute the topic-aware hidden state $\mathbf{r}_t$ by adding up $\mathbf{h}_t$ and $\mathbf{e}_t$, which serves as the initial hidden state of sentence decoder:

$$\mathbf{r}_t = \mathbf{h}_t + \mathbf{e}_t \quad (13)$$

Additionally, a stop confirmation is executed at each time step:

$$p_{stop} = \sigma(\text{linear}(\mathbf{h}_t)) \quad (14)$$

where σ represents the sigmoid function. If $p_{stop} > 0.5$, TWAG will terminate the decoding process and no more abstract sentences will be generated.

4.2.3 Sentence Decoder

Our sentence decoder adopts the Pointer-Generator network (See et al., 2017), which picks tokens both from input paragraphs and vocabulary.

To copy a token from the input paragraphs, the decoder requires the token-wise hidden states $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{n_u})$ of all n_u input tokens, which is obtained by concatenating the token-wise hidden states of all TTGs, i.e.,

$$\mathbf{U} = [\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_{n_u}] \quad (15)$$

For the k-th token, the decoder computes an attention distribution $\mathbf{a}_k$ over tokens in the input paragraphs, where each element $\mathbf{a}_k^i$ could be viewed as the probability of the i-th token being selected,

$$\mathbf{a}_k^i = \text{softmax}(\tanh(\mathbf{W}_u\mathbf{u}_i + \mathbf{W}_s\mathbf{s}_k + \mathbf{b}_a)) \quad (16)$$

where $\mathbf{s}_k$ denotes the decoder hidden state with $\mathbf{s}_0 = \mathbf{r}_t$ to incorporate the topic-aware representation, and $\mathbf{W}_u, \mathbf{W}_s, \mathbf{b}_a$ are trainable parameters.

To generate a token from the vocabulary, we first use the attention mechanism to calculate the weighted sum of encoder hidden states, known as the context vector,

$$\mathbf{c}_k^* = \sum_i \mathbf{a}_k^i \mathbf{u}_i. \quad (17)$$

which is further fed into a two-layer network to obtain the probability distribution over vocabulary,

$$P_{voc} = \text{softmax}(\text{linear}(\text{linear}([\mathbf{s}_k, \mathbf{c}_k^*]))). \quad (18)$$

To switch between these two mechanisms, p_{gen} is computed from context vector $\mathbf{c}_k^*$, decoder hidden state $\mathbf{s}_k$ and decoder input $\mathbf{x}_k$:

$$p_{gen} = \sigma(\mathbf{W}_c^T\mathbf{c}_k^* + \mathbf{W}_s^T\mathbf{s}_k + \mathbf{W}_x^T\mathbf{x}_k + \mathbf{b}_p) \quad (19)$$

where σ represents the sigmoid function and $\mathbf{W}_c^T, \mathbf{W}_s^T, \mathbf{W}_x^T$ and $\mathbf{b}_p$ are trainable parameters. The final probability distribution of words is[2]

$$P(w) = p_{gen}P_{voc}(w) + (1 - p_{gen}) \sum_{i:|ww_i=w} \mathbf{a}_k^i \quad (20)$$

4.3 Training

The modules for topic detection and abstract generation are trained separately.

4.3.1 Topic Detector Training

Since there are no public benchmarks for assigning input paragraphs with Wikipedia topics, we construct the dataset with existing Wikipedia articles. In each domain, we collect all the label-content pairs $\{(l, p)\}$ (defined in Section 3), and split the content into paragraphs $p = (d_1, d_2, \ldots, d_{n_p})$ to form a set of label-paragraph pairs $\{(l, d)\}$. Afterwards, we choose all pairs (l, d) whose section label l belongs to a particular topic $T \in \mathcal{T}$ to complete the dataset construction, i.e., the topic-paragraph set $\{(T, d)\}$. Besides, a *NOISE* topic is

[2] ww_i means the token corresponding to $\mathbf{u}_i$.

set up in each domain, which refers to meaningless text like scripts and advertisements, and the corresponding paragraphs are obtained by utilizing regular expressions to match obvious noisy texts. The details are reported in Appendix A.

Note that the dataset for abstract generation is collected from non-Wikipedia websites (refer to Section 5 for details). These two datasets are independent of each other, which prevents potential data leakage.

In the training step, we use the negative log-likelihood loss to optimize the topic detector.

4.3.2 Abstract Generator Training

The loss of topic-aware abstract generation step consists of two parts: the first part is the average loss of sentence decoder for each abstract sentence $\mathcal{L}_{sent}$, and the second part is the cross-entropy loss of stop confirmation $\mathcal{L}_{stop}$.

Following (See et al., 2017), we compute the loss of an abstract sentence by averaging the negative log likelihood of every target word in that sentence, and achieve $\mathcal{L}_{sent}$ via averaging over all m sentences,

$$\mathcal{L}_{sent} = \frac{1}{m}\sum_{t=1}^{m}\left(\frac{1}{n_{s_t}}\sum_{i=1}^{n_{s_t}} -\log P(w_i)\right) \quad (21)$$

where n_{s_t} is the length of the t-th sentence of the abstract. As for $\mathcal{L}_{stop}$, we adopt the cross-entropy loss, i.e.,

$$\mathcal{L}_{stop} = -y_s \log(p_{stop}) - (1 - y_s)\log(1 - p_{stop}) \quad (22)$$

where $y_s = 1$ when $t > m$ and $y_s = 0$ otherwise.

5 Experiments

5.1 Experimental Settings

Dataset. To evaluate the overall performance of our model, we use the **WikiCatSum** dataset proposed by (Perez-Beltrachini et al., 2019), which contains three distinct domains (*Company*, *Film* and *Animal*) in Wikipedia. Each domain is split into train (90%), validation (5%) and test (5%) set.

We build the dataset for training and evaluating the topic detector from the 2019-07-01 English Wikipedia full dump. For each record in the WikiCatSum dataset, we find the article with the same title in Wikipedia dump, and pick all section label-content pairs $\{(l, p)\}$ in that article. We remove all hyperlinks and graphics in contents, split the contents into paragraphs with the *spaCy* library, and follow the steps in Section 4.3.1 to complete dataset construction. Finally, we conduct an 8:1:1 split for train, validation and test.

Table 1 presents the detailed parameters of used datasets.

Evaluation Metrics. We evaluate the performance of our model with ROUGE scores (Lin, 2004), which is a common metric in comparing generated and standard summaries. Considering that we do not constrain the length of generated abstracts, we choose ROUGE F1 score that combines precision and recall to eliminate the tendency of favoring long or short results.

Implementation Details. We use the open-source *PyTorch* and *transformers* library to implement our model. All models are trained on NVIDIA GeForce RTX 2080.

In topic detection, we choose the top 20 frequent section labels in each domain and manually group them into different topics (refer to the Appendix A for details). For training, we use the pretrained *albert-base-v2* model in the *transformers* library, keep its default parameters and train the module for 4 epochs with a learning rate of 3e-5.

For abstract generation, we use a single-layer BiGRU network to encode the TTGs into hidden states of 512 dimensions. The first 400 tokens of input paragraphs are retained and transformed into GloVe (Pennington et al., 2014) embedding of 300 dimensions. The vocabulary size is 50000 and out-of-vocabulary tokens are represented with the average embedding of its adjacent 10 tokens. This module is trained for 10 epochs, the learning rate is 1e-4 for the first epoch and 1e-5 for the rest.

Before evaluation, we remove sentences that have an overlap of over 50% with other sentences to reduce redundancy.

Baselines. We compare our proposed **TWAG** with the following strong baselines:

- **TF-S2S** (Liu et al., 2018) uses a Transformer decoder and compresses key-value pairs in self-attention with a convolutional layer.
- **CV-S2D+T** (Perez-Beltrachini et al., 2019) uses a convolutional encoder and a two-layer hierarchical decoder, and introduces LDA to model topical information.
- **HierSumm** (Liu and Lapata, 2019) utilizes the attention mechanism to model inter-

Domain	#Examples	R1-r	R2-r	RL-r	#Topics	Train	Valid	Test
Company	62,545	.551	.217	.438	4	35,506	1,999	2,212
Film	59,973	.559	.243	.456	5	187,221	10,801	10,085
Animal	60,816	.541	.208	.455	4	51,009	2,897	2,876

Table 1: Details about used datasets. The left half shows parameters about the WikiCatSum dataset: number of examples and ROUGE 1, 2, L recalls. The right half shows parameters about the dataset for topic detector: number of topics and number of topic-paragraph pairs in each split.

Model	Company			Film			Animal		
	R1	R2	RL	R1	R2	RL	R1	R2	RL
TF-S2S	.197	.023	.125	.198	.065	.172	.252	.099	.210
CV-S2D+T	.275	.106	.214	.380	**.212**	.323	.427	.279	.379
HierSumm	.133	.028	.107	.246	.126	.185	.165	.069	.134
BART	.310	.116	.244	.375	.199	.325	.376	.226	.335
TWAG (ours)	**.341**	**.119**	**.316**	**.408**	**.212**	**.343**	**.431**	**.244**	**.409**

Table 2: ROUGE F1 scores of different models.

paragraph relations and then enhances the document representation with graphs.

- **BART** (Lewis et al., 2020) is a pretrained sequence-to-sequence model that achieved success on various sequence prediction tasks.

We fine-tune the pretrained BART-base model on our dataset and set beam size to 5 for all models using beam search at test time. The parameters we use for training and evaluation are identical to these in corresponding papers.

5.2 Results and Analysis

Table 2 shows the ROUGE F1 scores of different models. In all three domains, TWAG outperforms other baselines. Our model surpasses other models on ROUGE-1 score by a margin of about 10%, while still retaining advantage on ROUGE-2 and ROUGE-L scores. In domain *Company*, our model boosts the ROUGE-L F1 score by about 30%, considering that ROUGE-L score is computed upon the longest common sequence, the highest ROUGE-L score indicates that abstracts generated by TWAG have the highest holistic quality.

While CVS2D+T and BART retain reasonable scores, TF-S2S and HierSumm do not reach the scores they claim in their papers. Notice that the WikiCatSum dataset is a subset of WikiSum, which is used as the training dataset of these two models, we infer that TF-S2S and HierSumm require more training data to converge, and suffer from under-fitting due to the dataset scale. This phenomenon also proves that TWAG is data-efficient.

5.3 Ablation Study

Learning Rate of Topic Detector. We tried two learning rates when training the topic detector module. A learning rate of 1e-7 would result in a precision of 0.922 in evaluation, while a learning rate of 3e-5 would result in a precision of 0.778. However, choosing the former learning rate causes a drop of about 10% in all ROUGE scores, which is the reason why we use the latter one in our full model.

We infer that human authors occasionally make mistakes, assigning paragraphs into section labels that belong to other topics. A topic detector with low learning rate overfits these mistakes, harming the overall performance of our model.

Soft or Hard Topic. To further investigate the effectiveness of TWAG's soft topic mechanism, we compare the results of soft and hard topic and report them in Table 4, from which we can see that hard topic does quite poorly in this task.

Topic Detector	Hard Topic			Soft Topic		
	R1	R2	RL	R1	R2	RL
Company	.266	.074	.245	**.341**	**.119**	**.316**
Film	.355	.159	.333	**.408**	**.212**	**.343**
Animal	.407	.223	.387	**.431**	**.244**	**.409**

Table 4: ROUGE F1 scores of different topic selectors.

A possible reason is that some sentences in the

Gold Abstract: manjina there (kannada : ⟨unk⟩ ⟨unk⟩) is a 1980 indian kannada film, directed by bangalore nagesh and produced by k. r. narayana murthy. the film stars srinath, manjula, thoogudeepa srinivas and dinesh in lead roles. the film had musical score by upendra kumar.
TF-S2S: chalo ishq larain is a pakistani film directed by sajjad gul and was released across pakistan in april 2002 starring meera , ali haider and zara sheikh.
CV-S2D+T: ⟨unk⟩ ⟨unk⟩ (kannada : ⟨unk⟩ ⟨unk⟩) is a 1967 indian kannada film, directed by b. mallesh and produced by b. s. ranga. the film stars rajkumar, udaykumar, narasimharaju and k. s. ashwath in lead roles. the film had musical score by chellapilla satyam . the film was a remake of hindi film ⟨unk⟩.
HierSumm: (kannada : ⟨unk⟩ ⟨unk⟩) is a 1980 indian kannada film , directed by bangalore nagesh . the film stars srinath , manjulla , thoogudeepa shreenivaas . the film stars srinath , manjula , manjula , thoogudeepa shreenivaas in lead roles . the film had musical score by upendra kumar . the film is a remake of telugu movie aakashagiri rao . the movie was remade in telugu as manjina in 1986 . the movie was remade in telugu as manjina there . . .
BART: manjina there is a 1980 kannada family drama film directed by bangalore nagesh starring srinath and manjula in the lead roles. it was released on 14 january 1980.
TWAG: manjina there is a 1980 kannada drama film directed by bangalore nagesh. the film stars srinath, vajramuni, manjula and thoogudeepa srinivas in lead roles. the film had musical score by upendra kumar and the film opened to positive reviews in 1980. the film was a remake of tamil film ⟨unk⟩.

Table 3: Comparison between Wikipedia abstracts generated by different models about the film *Majina There*. Non-English characters have been replaced with ⟨unk⟩ for readability.

standard abstract express more than one topic. Assigning one topic to each sentence will result in semantic loss and thus harm the quality of generated abstract, while the soft topic could better simulate the human writing style.

Number of Section Labels. The number of section labels n_t plays a key role in our model: a small n_t would not be informative enough to build topics, while a large one would induce noise. We can see from Figure 3 that the frequency of section labels is long-tailed, thus retaining only a small portion is able to capture the major part of information. Table 5 records the experiment results we conducted on domain *Company*. $n_t = 20$ reaches a peak on ROUGE 1, 2 and L scores, indicating that 20 is a reasonable number of section labels.

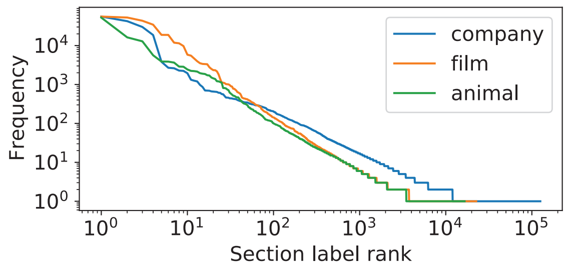


Figure 3: The frequency of section labels in three domains. When ignoring section labels with extra high or low frequency, remaining section labels' frequency and rank generally form a straight line in log scale, which matches the Zipf's law for long-tail distributions.

#Labels	R1	R2	RL
10	.337	.117	.312
20	**.340**	**.118**	**.315**
30	.336	.117	.311

Table 5: ROUGE F1 scores of different n_t.

5.4 Case Study

Table 3 shows the generated Wikipedia abstracts by different models about film *Majina There*. We can see that the gold abstract contains information about three topics: basic information (region, director, and producer), actors, and music.

Among the models, TF-S2S produces an abstract with a proper pattern but contains wrong information and BART misses the musical information topic. CV-S2D+T, HierSumm, and our TWAG model both cover all three topics in the gold abstract, however, CV-S2D+T makes several factual errors like the release date and actors and HierSumm suffers from redundancy. TWAG covers all three topics in the gold abstract and discovers extra facts, proving itself to be competent in generating comprehensive abstracts.

5.5 Human Evaluation

We follow the experimental setup of (Perez-Beltrachini et al., 2019) and conduct a human evaluation consisting of two parts. A total of 45 examples (15 from each domain) are randomly selected from the test set for evaluation.

The first part is a question-answering (QA) scheme proposed in (Clarke and Lapata, 2010) in order to examine factoid information in summaries. We create 2-5 questions[3] based on the golden sum-

[3]Example questions are listed in the Appendix C, and the whole evaluation set is included in the our code repository.

Model	Company		Film		Animal	
	Score	Non-0	Score	Non-0	Score	Non-0
TF-S2S	.075	.694	.000	.000	.000	.000
CV-S2D+T	.237	.660	.040	.143	.382	.576
HierSumm	.255	.896	.213	.327	.000	.000
BART	.591	.813	.452	.796	.342	.653
TWAG (ours)	**.665**	**.903**	**.669**	**.918**	**.543**	**.868**

Table 6: Human evaluation results in QA scheme. Score represents the mean score and non-0 represents the percentage of answered questions.

Model	Company			Film			Animal		
	C	F	S	C	F	S	C	F	S
TF-S2S	2.69	2.71	2.67	1.93	2.71	2.84	2.22	2.96	2.76
CV-S2D+T	2.42	2.36	2.73	2.29	2.69	2.98	2.80	3.18	3.18
HierSumm	**2.96**	2.64	1.69	3.13	2.78	2.04	2.80	3.13	1.82
BART	2.64	2.82	**3.00**	2.87	3.02	3.24	2.78	3.11	3.00
TWAG (ours)	2.91	**2.87**	2.91	**3.20**	**3.16**	**3.44**	**3.56**	**3.58**	**3.40**

Table 7: Human evaluation results in linguistic quality scoring. C indicates completeness, F indicates fluency and S indicates succinctness.

mary which covers the appeared topics, and invite 3 participants to answer the questions by taking automatically-generated summaries as background information. The more questions a summary can answer, the better it is. To quantify the results, we assign a score of 1/0.5/0.1/0 to a correct answer, a partially correct answer, a wrong answer and those cannot be answered, and report the average score over all questions. Notice that we give a score of 0.1 even if the participants answer the question incorrectly, because a wrong answer indicates the summary covers a certain topic and is superior to missing information. Results in Table 6 shows that 1) taking summaries generated by TWAG is capable of answering more questions and giving the correct answer, 2) TF-S2S and HierSumm perform poorly in domain *Film* and *Animal*, which is possibly a consequence of under-fitting in small datasets.

The second part is an evaluation over linguistic quality. We ask the participants to read different generated summaries from 3 perspectives and give a score of 1-5 (larger scores indicates higher quality): **Completeness** (does the summary contain sufficient information?), **Fluency** (is the summary fluent and grammatical?) and **Succinctness** (does the summary avoid redundant sentences?) Specifically, 3 participants are assigned to evaluate each model, and the average scores are taken as the final results. Table 7 presents the comparison results, from which we can see that, the linguistic quality of TWAG model outperforms other baseline models, validating its effectiveness.

6 Conclusion

In this paper, we propose a novel topic-guided abstractive summarization model TWAG for generating Wikipedia abstracts. It investigates the section labels of Wikipedia, dividing the input document into different topics to improve the quality of generated abstract. This approach simulates the way how human recognize entities, and experimental results show that our model obviously outperforms existing state-of-the-art models which view Wikipedia abstracts as plain text. Our model also demonstrates its high data efficiency. In the future, we will try to incorporate pretrained language models into the topic-aware abstract generator module, and apply the topic-aware model to other texts rich in topical information like sports match reports.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. This work is supported by the National Key Research and Development Program of China (2017YFB1002101), NSFC Key Project (U1736204) and a grant from Huawei Inc.

Ethical Considerations

TWAG could be applied to applications like automatically writing new Wikipedia abstracts or other texts rich in topical information. It can also help human writers to examine whether they have missed information about certain important topics.

The benefits of using our model include saving human writers' labor and making abstracts more comprehensive. There are also important considerations when using our model. Input texts may violate copyrights when inadequately collected, and misleading texts may lead to factual mistakes in generated abstracts. To mitigate the risks, researches on how to avoid copyright issues when collecting documents from the Internet would help.

References

Siddhartha Banerjee and Prasenjit Mitra. 2016. Wikiwrite: generating wikipedia articles automatically. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 2740–2746.

Siddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. 2015. Multi-document abstractive summarization using ilp based multi-sentence compression. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 1208–1214.

Fadi Biadsy, Julia Hirschberg, and Elena Filatova. 2008. An unsupervised approach to biography production using wikipedia. In *Proceedings of ACL-08: HLT*, pages 807–815.

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca J Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1587–1597.

Xiaoyan Cai and Wenjie Li. 2012. Mutually reinforced manifold-ranking based relevance propagation model for query-focused multi-document summarization. *IEEE transactions on audio, speech, and language processing*, 20(5):1597–1607.

James Clarke and Mirella Lapata. 2010. Discourse constraints for document compression. *Computational Linguistics*, 36(3):411–441.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479.

Alexander Richard Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084.

Katja Filippova and Michael Strube. 2008. Sentence fusion via dependency graph compression. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 177–185.

Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109.

Hiroaki Hayashi, Prashant Budania, Peng Wang, Chris Ackerson, Raj Neervannan, and Graham Neubig. 2021. Wikiasp: A dataset for multi-domain aspect-based summarization. *Transactions of the Association for Computational Linguistics*, 9:211–225.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. 2016. Wikireading: A novel large-scale language understanding task over wikipedia. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1535–1545.

Luyang Huang, Lingfei Wu, and Lu Wang. 2020. Knowledge graph-augmented abstractive summarization with semantic-driven cloze reward. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5094–5107.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Wei Li, Xinyan Xiao, Jiachen Liu, Hua Wu, Haifeng Wang, and Junping Du. 2020. Leveraging graph to improve abstractive multi-document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6232–6243.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*.

Yang Liu and Mirella Lapata. 2019. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv*, pages arXiv–1907.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing)*, pages 1532–1543.

Laura Perez-Beltrachini, Yang Liu, and Mirella Lapata. 2019. Generating summaries with topic templates and structured convolutional decoders. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5107–5116.

Dragomir Radev. 2000. A common theory of information fusion from multiple text sources step one: cross-document structure. In *1st SIGdial workshop on Discourse and dialogue*, pages 74–83.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Christina Sauper and Regina Barzilay. 2009. Automatically generating wikipedia articles: a structure-aware approach. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 208–216.

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.

Xiaojun Wan. 2008. An exploration of document impact on graph-based multi-document summarization. In *Proceedings of the 2008 conference on empirical methods in natural language processing*, pages 755–762.

Xiaojun Wan and Jianguo Xiao. 2009. Graph-based multi-modality learning for topic-focused multi-document summarization. In *Proceedings of the 21st international jont conference on Artifical intelligence*, pages 1586–1591.

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*, pages 452–462.

Yongjing Yin, Linfeng Song, Jinsong Su, Jiali Zeng, Chulun Zhou, and Jiebo Luo. 2019. Graph-based neural sentence ordering. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 5387–5393.

Jianmin Zhang, Jiwei Tan, and Xiaojun Wan. 2018. Towards a neural network approach to abstractive multi-document summarization. *arXiv preprint arXiv:1804.09010*.

A Topic Allocation

For each domain, we sort section labels by frequency and choose the top $n_t = 20$ frequent section labels, then manually allocate them into different topics. Section labels with little semantic information like *Reference* and *Notes* are discarded in allocation to reduce noise. Table 8 shows how we allocate section labels into topics in domain *Company*, *Film* and *Animal*.

An additional *NOISE* topic is added to each domain to detect website noises. We build training records for *NOISE* by finding noise text in the training set of WikiCatSum by regular expressions. For example, we view all text containing "cookie", "href" or text that seems to be a reference as noise.

B Trivia about Baselines

We use BART-base as the baseline for comparison because BART-large performs poorly in experiments. BART-large starts generating redundant results when using only 4% training data, and its training loss also decreases much slower than

BART-base. We infer that BART-large may overfit on training data, and BART-base is more competent to be the baseline.

C Human Evaluation Example

Table 9 shows an example of gold summary, its corresponding question set and system outputs. The full dataset we used for human evaluation can be found in our code repository.

Domain	Topic	Section Labels
Company	History	History, Company history, Ownership
	Product	Products, Services, Destinations, Products and services, Technology
	Location	Fleet, Operations, Subsidiaries , Locations
	Reception	Awards, Controversies, Controversy, Criticism, Accidents and incidents, Reception
Film	Cast	Cast, Casting
	Plot	Plot, Synopsis, Plot summary
	Production	Production, Filming, Development
	Reception	Reception, Critical reception, Critical response, Awards, Accolades, Awards and nominations
	Box office	Box office
Animal	Distribution	Distribution
	Taxonomy	Taxonomy, Species, Subspecies, Classification
	Description	Description, Habitat, Ecology, Behaviour, Biology, Diet, Feeding, Breeding, Reproduction, Life cycle
	Conservation Status	Status, Conservation, Conservation status

Table 8: Detailed allocation of section labels. In domain *Company*, Orange labels are labels selected when $n_t = 10$, Green labels are additional labels selected when $n_t = 20$, and Blue labels are additional labels selected when $n_t = 30$.

Gold Abstract: l'avare is a 1980 french comedy film written and directed by louis de funès and jean girault , and starring de funès. the english title of the film is the miser. it is an adaptation of molière 's famous comedy l'avare ou l'école du mensonge (the miser). de funès tried to draw out the unhappy side of the character. harpagon , unloved by humanity, is driven to an obsessive love of money.
Questions: 1. When and where was l'avare released? 2. Who stars in l'avare? 3. Who directed l'avare? 4. What is the English name of l'avare?
TF-S2S: der er et yndigt land is a 1983 danish drama film directed by morten arnfred. it was entered into the 33rd berlin international film festival , where it won an honourable mention.
CV-S2D+T: &lt;unk&gt; 's &lt;unk&gt; is a french comedy film from 1954, directed by jean girault, starring jean marais and louis de funès. it was screened in the un certain regard section at the 2015 cannes film festival.
HierSumm:(hangul ; rr : l'am) is a french drama film directed by louis de funès. it is based on a play by molière. it stars louis de funès. it was entered into the 36th berlin international film festival. the film was nominated for the golden globe award for best foreign language film. it was also nominated for the golden globe award for best foreign language film. . . .
BART: l'avare (english : the miser) is a 1980 french drama film directed by louis de funès and starring jean girault. it was based on an original screenplay co-written with julien françois .
TWAG: the miser (french : l'avare) is a 1980 french drama film directed by funes de funès. the film stars louis de funès , sanctioning cléante , and broach harpagon.

Table 9: Example of Gold summary, question set and system outputs for the QA evaluation study.