Unsupervised Single Document Abstractive Summarization using Semantic Units

Jhen-Yi Wu and **Ying-Jia Lin** and **Hung-Yu Kao**
Intelligent Knowledge Management Lab
Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan
jhenyiwu.d@gmail.com, yingjia.lin.public@gmail.com,
hykao@mail.ncku.edu.tw

Abstract

In this work, we study the importance of content frequency on abstractive summarization, where we define the content as "semantic units." We propose a two-stage training framework to let the model automatically learn the frequency of each semantic unit in the source text. Our model is trained in an unsupervised manner since the frequency information can be inferred from source text only. During inference, our model identifies sentences with high-frequency semantic units and utilizes frequency information to generate summaries from the filtered sentences. Our model performance on the *CNN/Daily Mail* summarization task outperforms the other unsupervised methods under the same settings. Furthermore, we achieve competitive ROUGE scores with far fewer model parameters compared to several large-scale pre-trained models. Our model can be trained under low-resource language settings and thus can serve as a potential solution for real-world applications where pre-trained models are not applicable.

1 Introduction

Summarization is a task involving compressing a longer text into a shorter version while preserving the salient information in the original text. When given article-summary pairs, supervised models are able to learn corresponding implicit relationships, for example, where to focus or what to preserve. However, a lack of sufficient training pairs is a common issue in real-world applications. Creating such high-quality training pairs can be costly. Although large pre-trained models for language generation or summarization may require less data for fine-tuning, they are often trained on English corpus only (e.g., Raffel et al., 2020; Song et al., 2019; Lewis et al., 2020; Zhang et al., 2020) and thus are not suitable for low-resource languages. Therefore, we seek the possibility of unsupervised summarization methods.

Our idea is to utilize the frequency of contents in the source text. Intuitively, we expect some specific contents to be included in a summary if they frequently occur in the source article. A similar concept of "content units" was first proposed by Nenkova and Passonneau (2004). They manually labeled the text by identifying similar text segments to form a content unit, where the contributing text segments of a content unit should have similar semantic meanings. In their results (Nenkova and Vanderwende, 2005), of the top 5 most frequent content units in the source documents, 96% appear in a human summary, and high percentages of 92 and 85 are observed for the top 8 and top 12 most frequent content units across 11 input sets. Their observation shows that content unit frequency can provide huge hints as to whether a specific unit of content will be selected as a part of a human-written summary and therefore supports our idea. We also provide our statistical results on the recent summarization dataset *CNN/Daily Mail* (See et al., 2017) in Appendix A.2.

Instead of manually labeling content units like Nenkova and Passonneau (2004), we divide and enumerate all text spans with a fixed-size sliding window. Here, we refer to the divided text spans as "semantic units" (SUs), as we expect each semantic unit to contain brief semantic concepts in itself. We then argue that a refined summary should at least contain the semantic units frequently occurring in the original articles since the high-frequency semantic units should be the topic or contain key descriptions. In addition, frequency information alone should be possible to retrieve from source documents only. In this work, we propose a model that automatically learns semantic unit frequency. The learned frequency information is then used to discriminate salient parts in source documents for abstractive summarization.

In our proposed method, which is shown in Figure 1, the training process is divided into two

Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 954–966
November 20–23, 2022.

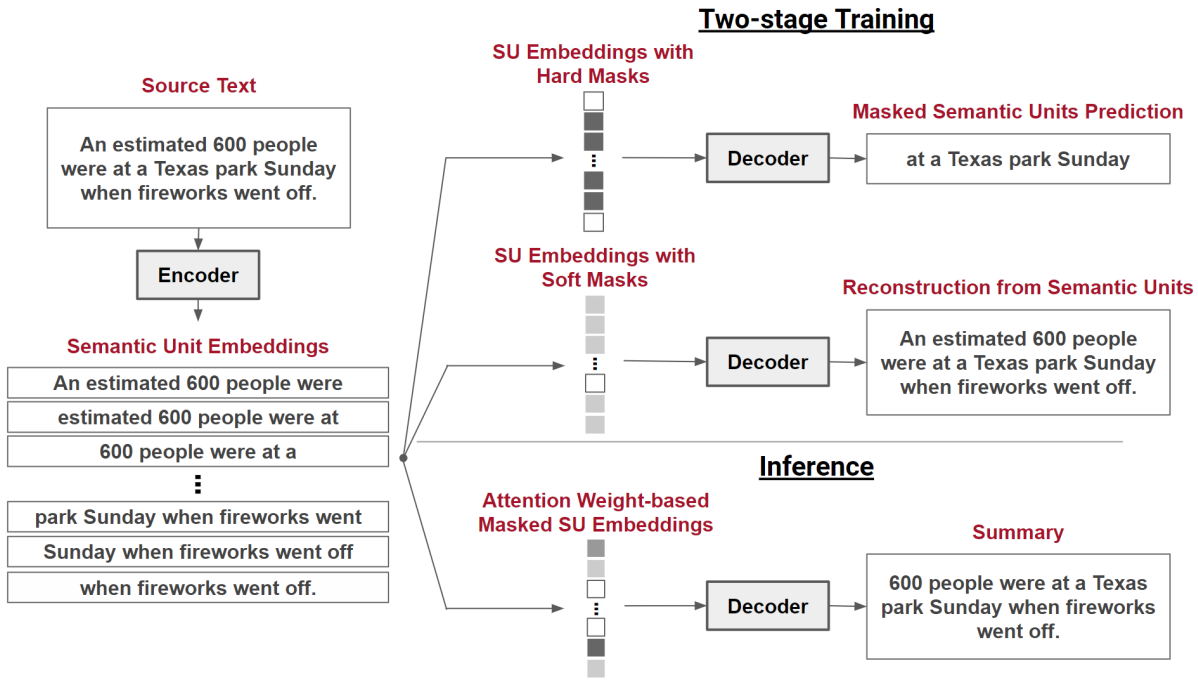


Figure 1: Our training and inference stages. The semantic unit embeddings with darker colors indicate that greater attention mask values are applied.

stages. In the first training stage, our model learns to predict the masked tokens based on the partially masked semantic units. This stage mimics the masked language modeling objectives used in the pre-trained language models (Devlin et al., 2019; Song et al., 2019; Lewis et al., 2020). In the second training stage, the training goal of the model is to generate fluent text based on the given semantic units. We train the model to reconstruct the original articles in this stage; thus, no human-written summaries are used during training. In the inference stage, semantic unit frequency is obtained using the attention mechanism, which helps the model decide how much to focus on the semantic units when generating text. We first let the model generate text based on all semantic units in a given article and record the attention weights for each semantic unit. The recorded attention weights are used to assign weights to the semantic units. The weights are considered the semantic unit frequency since they represent how much the model has focused on each semantic unit when reconstructing the original article. The weighted semantic units are used to filter the sentences in the source text, and the corresponding weighted semantic units are provided to the decoder to generate a sequence. The generated sequence is considered the final summary.

Here, we list our contributions: First, our experiments prove that our proposed model discriminates semantic units by frequency and generates summaries from them. Second, our model parameters are far fewer than many other pre-trained models, but we can still achieve competitive ROUGE scores. Finally, no single summary is used in our training and inference process; therefore, our proposed method is suitable for real-world applications where human-written summaries are rarely accessible. Our code is publicly available at https://github.com/IKMLab/UASSU.

2 Related Work

Sentence compression. Sentence compression can be seen as a small-scale text summarization task. Most earlier work focused on removal of unnecessary words (Knight and Marcu, 2002; Dorr et al., 2003). Since neural network-based approaches have been proposed, recent works utilize sequence-to-sequence models to solve this task (Févry and Phang, 2018; Baziotis et al., 2019; Zhou and Rush, 2019). In these approaches, the goals of compressing or contextual matching may not be suitable for long text summarization, where the summaries are not expected to be contextually similar to the entire content of the original articles, and compression is not adequate to remove detailed descriptions in a long text. This tendency is also shown in Févry and Phang's (2018) experiments, where they discovered that the length of input sentences also affects

model performance, suggesting that directly applying sentence compression methods on longer text summarization tasks is challenging.

Text summarization. Studies on longer text inputs and more general cases for summarization have since been discovered. Dohare et al. (2018) provided an Abstract Meaning Representation-based (AMR) solution, but it requires an extra AMR-to-text model, where the corresponding training data is unlikely to be accessible for low-resource languages. Laban et al. (2020) utilized a reinforcement learning-based model to generate summaries that can be used to better recover the keywords in source documents. They fine-tuned two large-scale pre-trained models, BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019), for modeling coverage and fluency of the generated summaries, respectively. Wang and Lee (2018) proposed a novel framework that used generative adversarial networks (GAN) to achieve unsupervised abstractive summarization. Their approach was based on the idea that, given an input document, the generator should try to generate shorter text that is readable by human and provides sufficient information that can be used by the reconstructor to reconstruct the original document. They utilized the discriminator in the GAN structure to determine if the generated text is human-readable or machine-generated. Their solution requires no additional data or any pre-trained models. It therefore suits our defined setting the most, where the solutions should not be constrained to large pre-training corpora or paired data. Other text summarization approaches differ in terms of the target domains, for example, review summarization (Isonuma et al., 2019), meeting speech summarization (Shang et al., 2018), and five-sentence story summarization (Liu et al., 2019), or focuses on multi-document settings (Chu and Liu, 2019; Bražinskas et al., 2020). These approaches often utilize specific techniques or assumptions for various targeted domains.

Zero-shot pre-training. Recent works have utilized large-scale pre-trained models to achieve zero-shot abstractive summarization (Zhu et al., 2019; Yang et al., 2020; Fabbri et al., 2021). For example, in Yang et al.'s (2020) work, they leveraged the so-called "lead bias" characteristic to create a large amount of paired data from news data collected online. Lead bias is a well-known characteristic in recent summarization datasets. It means that extracting the first few sentences alone as summaries can yield fair performance in terms of the ROUGE scores and can even outperform many sophisticated summarization models. Yang et al. (2020) directly utilized this characteristic to generate pseudo summaries and used them to pre-train their model. In the fine-tuning process, they used a denoising autoencoder and theme modeling to enhance the model performance. On the other hand, Fabbri et al. (2021) created pseudo article-summary paired data from Wikipedia as the fine-tuning data for pre-trained language generation models. Then they grouped the pseudo paired data by abstractiveness. For each target dataset, they used the paired data of corresponding abstractiveness for fine-tuning. They proved that improvements could be made in zero-shot domain transfer and few-shot settings through Wikipedia data fine-tuning. However, lead bias may not be observed in all kinds of datasets in different domains or languages, suggesting that more general solutions should be discovered.

As novel approaches are proposed, one can see that the trend also implies that current methods favor using large-scale pre-trained models, which obviously ignore the needs under specific scenarios where training data is difficult to obtain. In contrast, our proposed approach provides a training regime that does not require any pre-trained models or massive amounts of paired data.

3 Method

We briefly introduce the model structure and semantic unit construction in Section 3.1. Next, in Section 3.2, we divide our training strategy into two stages and describe them separately. Finally, we explain how we leverage the learned frequency information for unsupervised text summarization in Section 3.3. The overview of training and inference stages is also shown in Figure 2.

3.1 Semantic unit construction

We use standard Transformer encoder-decoder (Vaswani et al., 2017) as our model architecture. Each document is taken as an input sequence, and each sequence is then tokenized into a list of tokens, $\mathbf{w} = \{w_0, w_1, ..., w_{n-1}\}$, where n is the length of the input sequence. The Transformer encoder encodes the input sequence, $\mathbf{w}$, into token embeddings, $\mathbf{h}$, with an embedding size d_h. The semantic units are constructed with the following steps: We first divide $\mathbf{h}$ with a sliding window (size c and stride s). In our experiments, the value of s is set to

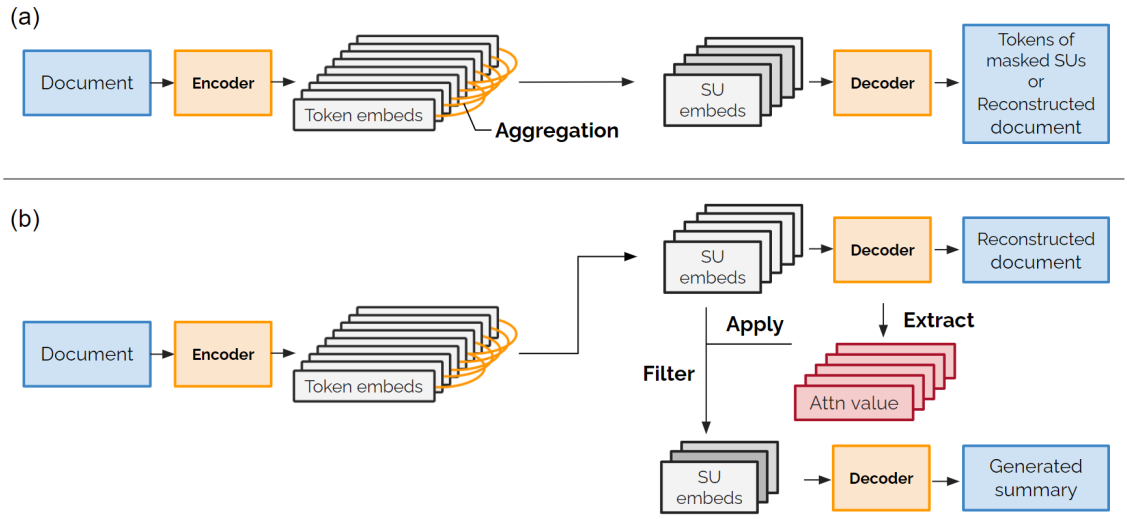


Figure 2: Our model overview. (a) The two-stage training process. (b) The inference process.

1 to enumerate all possible semantic units. Then we average[1] the token embeddings within each window to construct a semantic unit embedding. We denote the obtained semantic unit embeddings as $\mathbf{z}$ with an embedding size d_z. Here d_z is equal to d_h.

3.2 Two-stage training

3.2.1 Masked semantic units prediction

This section describes the first training stage, which helps our model learn to focus on the context in the source documents. To achieve this goal, we adjust the learning objectives used in various self-supervised models (Devlin et al., 2019; Song et al., 2019; Lewis et al., 2020) and customize the masked language modeling for our model to predict the masked semantic units. Instead of directly masking out the input tokens, which is how the previous studies did (Devlin et al., 2019; Song et al., 2019; Lewis et al., 2020), the masking unit here is a semantic unit embedding. We apply attention masks with the value β and the masking rate p_{mask} to the semantic unit embeddings $\mathbf{z}$. We refer to the attention masks as hard masks in this stage because β is a larger value compared to the next stage of training; therefore, the model cannot attend to the masked semantic units. The surrounding semantic units can hold information retained from shared tokens in the targeted semantic unit. Therefore, we ensure the surrounding semantic unit embeddings which share the tokens of the masked one are also masked. We denote $\mathbf{m} \subseteq \{0, 1, \ldots, n-1\}$ as the corresponding token indexes of the masked semantic units. To let our model focus on recovering masked semantic units only, we set the loss weight α close to 1 for each token of index $i \in \mathbf{m}$, which is shown in Equation 1. Therefore, the loss for the unmasked tokens in our objective function (Equation 2) is relatively small compared to that of the masked ones, implying that the unmasked tokens do not have to be predicted correctly.

$$weight_{i,0\leq i<n} = \begin{cases} \alpha & \text{if } i \in \mathbf{m}, \\ (1-\alpha) & \text{otherwise.} \end{cases} \quad (1)$$

$$loss_i = -\log\left(\frac{\exp(P(w_i))}{\sum_{j\in|Vocab|}\exp(P(w_j))}\right) * weight_i \quad (2)$$

3.2.2 Reconstruction from Semantic Units

An abstractive summarization model should produce fluent text sequences as final outputs. Therefore, in this stage, our training goal is to let our model generate a fluent paragraph by learning to reconstruct the original input documents. This is achieved by adjusting the following parameters:

- The value β of attention masks is decreased, as these will be calculated based on the learned frequency to represent weights for each semantic unit in the inference stage.
- The loss weight α is decreased, as we do not require the model to reconstruct the exact tokens for the masked positions since the masked semantic units should be less important.

[1] We provide experiments for different aggregation methods and window sizes in Appendix A.3.

- The length of input sequences n is decreased for faster training speed, and the number of input semantic units for the decoder will also be reduced due to the sentence filtering during inference.

- The masking rate p_{mask} is increased because a relatively small portion of the semantic units in the source should be focused on when summarizing.

3.3 Utilize learned frequency during inference

In the inference stage, we hope to let the model recognize semantic unit frequency and generate a condensed version of the source text based on them. To achieve this goal, we designed a procedure where we run the decoder in two rounds to extract the learned frequency and generate a summary based on the information. We describe the two rounds of decoding in this section and show them in Figure 2 (b).

First, we input a complete source document to the encoder and obtain semantic unit embeddings. In the first round of decoding, we provide all semantic unit embeddings to the decoder, and the decoder should reconstruct the source document as shown in the upper part of Figure 2 (b). We record the attention distribution in the second attention sublayer of the Transformer decoder (Vaswani et al., 2017) for each semantic unit embedding over all the decoding steps during reconstruction. The summation of each semantic unit's attention weights is considered the learned frequency information. If the model focuses more on a specific semantic unit when reconstructing the source text, that should mean the semantic unit is related to multiple parts in the original article. Therefore we expect the semantic units frequently mentioned in a source article to have a higher sum of attention weights than those appearing only a few times. Then we perform sentence filtering in each article based on the attention weights of the semantic units within each sentence. We select the sentences with the highest averaged attention scores of contained semantic units until the number of tokens in the selected sentences exceeds the value t. Finally, the semantic unit embeddings corresponding to the selected sentences are used to generate summaries in the next round of decoding.

The lower part of Figure 2 (b) shows the process for generating summaries. Before the second round of decoding, to let the model discriminate semantic unit frequency, we apply a value β for attention masks to the semantic units. The attention masks are computed based on the attention weights, and the masks are applied to the corresponding semantic units. Empirically, the value β of the attention mask for each semantic unit is computed by dividing the corresponding summation of attention weights by a constant λ ($\lambda = 100$). With the conversion, β for the semantic units with high attention weights should be large, and β should be a small value for the semantic units with low attention weights. Therefore, the masks serve as the weights on the semantic unit inputs, providing frequency information of each semantic unit to the decoder. The generated sequence based on the given weighted semantic units is considered the final summary.

4 Experiments

4.1 Settings

For our model structure, we use two layers each for the Transformer encoder and decoder. More Transformer layers are also applicable, and we leave the experiment in our future work. For both the encoder and decoder, we set 768 as the embedding size, 1024 as the feedforward embedding size, 8 heads for multi-head attention layers, and 0.1 for the dropout rate. For semantic unit construction, we set the sliding window size c at 5, and the stride s is set as 1. We use top-k sampling as the decoding strategy, where k is set at 5, for more abstractive summaries (Holtzman et al., 2020) and faster decoding speed than beam search. The minimum number of the tokens in the selected sentences in the inference stage, t, is set as 200. The desired length l for the generated summaries is set as 50 for *CNN/Daily Mail*, and the sentences that exceed l will be truncated. The other training configurations are listed as follows: 1e-4 for the learning rate, 3 for the maximum gradient clipping norm, and 4 for the batch size. Training took 6 to 8 hours per epoch on a GTX 1080 GPU. A pre-trained BERT-base-uncased tokenizer (Devlin et al., 2019) is used for tokenization. In each subsequent experiment, the models compared were all under the same settings and were trained with an equal number of steps.

4.2 Training strategies

During the two-stage training (Section 3.2), we trained our model for 16 and 12 epochs in the first and second stages. The second stage was further

Models	R1	R2	RL	# of Data	# of Model Parameters
Lead-3 Baseline (See et al., 2017)	40.34	17.70	36.57	-	-
Large scale pre-training or using pre-trained models					
Summary Loop 45 (Laban et al., 2020)	37.70	14.80	34.70	CNN/DM 280k articles	344M
Pegasus - Zero-shot (Zhu et al., 2019)	32.90	13.28	29.38	HugeNews (CNN/DM is included), 3.8 TB data	568M
BART-large - Zero-shot (Zhu et al., 2019)	32.83	13.30	29.64	Wikipedia+BookCorpus, 160 GB data	370M
T5 - Zero-shot (Zhu et al., 2019)	39.68	17.24	36.28	C4, 750 GB data	11B
Lead Bias Pre-training or Fine-tuning					
TED (Yang et al., 2020)	38.73	16.84	35.40	21.4 M news	370M
WikiTransfer (Fabbri et al., 2021)	39.11	17.25	35.73	60k Wikipedia articles, fine-tune on BART-large	370M
Bart-large-LB (Zhu et al., 2019)	40.52	17.63	36.76	21.4 M news, fine-tune on BART-large	370M
No paired data & No pre-training					
Unsupervised GAN - WGAN (Wang and Lee, 2018)	35.14	9.43	21.04	CNN/DM 280k articles	
Unsupervised GAN - Adversarial REINFORCE (Wang and Lee, 2018)	35.51	9.38	20.98	CNN/DM 280k articles	
Unsupervised GAN - Adversarial REINFORCE*	31.15	9.26	27.40	CNN/DM 280k articles	27M
Ours	**37.54**	**14.49**	**33.52**	CNN/DM 280k articles	**41M**

* Reimplemented by ourselves using the code provided by (Wang and Lee, 2018).

Table 1: Our ROUGE F_1 scores on the *CNN/Daily Mail* test set and their counterparts. R1, R2 and RL are the ROUGE-1, ROUGE-2 and ROUGE-L F_1 scores, respectively.

divided into 3 phases in practice, where our model was trained for 4 epochs in each phase during the second stage. As a result, there are 4 phases for the entire training, including the one in the first-stage training. For each phase, we truncated the article from 500, 400, 300, to 200 tokens for the input sequence length n, decreased the value of attention masks β from 1e+10, 1e+5, 1e+2, to 1e-1, decreased the loss weight α from 0.995, 0.95, 0.8 to 0.75, and increased the masking rate p_{mask} from 0.15 in the first phase and 0.30 for the following phases. Ablation studies about the two-stage training strategy and the inference workflow are provided in Appendix A.4 and A.5.

5 Results

5.1 ROUGE scores

For evaluating the proposed method, we use the non-anonymized version of *CNN/Daily Mail* (See et al., 2017; Hermann et al., 2015), where all named entities are retained in the source articles. Our results on *CNN/Daily Mail* are presented in Table 1.

For the comparison with the methods under the same unsupervised setting without massive pre-training, our model's scores exceed the ones in Wang and Lee's work by +2.03 ROUGE-1, +5.11 ROUGE-2, and +12.54 ROUGE-L points. Our ROUGE[2] scores are also much better than that of our reimplemented version of their model (Wang and Lee, 2018) (+6.39 ROUGE-1, +5.23 ROUGE-2, and +6.12 ROUGE-L points). In short, our model achieves the best results on unsupervised abstractive summarization when no paired data or pre-trained models are available. We also provide human evaluation results on Wang and Lee's work and ours in Appendix A.1.

In comparison to the zero-shot pre-training models, Pegasus (Zhang et al., 2020) and BART-large (Lewis et al., 2020), which were respectively pre-trained on 3.8 TB data and 160 GB data, our model trained with only *CNN/Daily Mail* 280k articles still exceeds their best scores by +4.64 ROUGE-1, +1.19 ROUGE-2, and +3.88 ROUGE-L. We observe a larger performance gap between our model and T5 (Raffel et al., 2020), which is an overwhelmingly large-scale model. However, there is a large difference in the number of parameters used in our model and T5. We use only 41M parameters which is much smaller than the 11B parameters of T5. Our model performance is comparable to Summary Loop 45's (Laban et al., 2020), which utilizes large-scale pre-trained models for their summarization system.

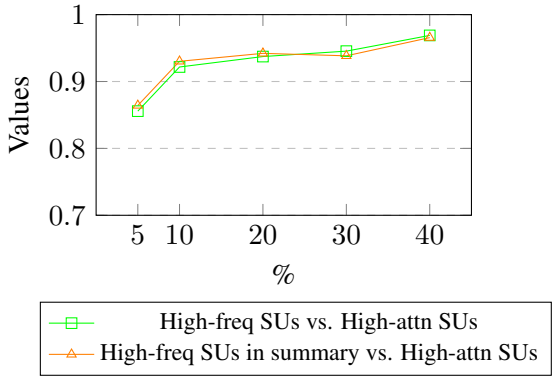
The models trained with pseudo paired data like TED (Yang et al., 2020), WikiTransfer (Fabbri et al., 2021), and BART-large-LB (Zhu et al., 2019) achieve inarguably better results than the scores of our model. However, considering the total data usage and model sizes, our method is more applicable for obtaining quicker and equivalent results

[2]https://github.com/bheinzerling/pyrouge

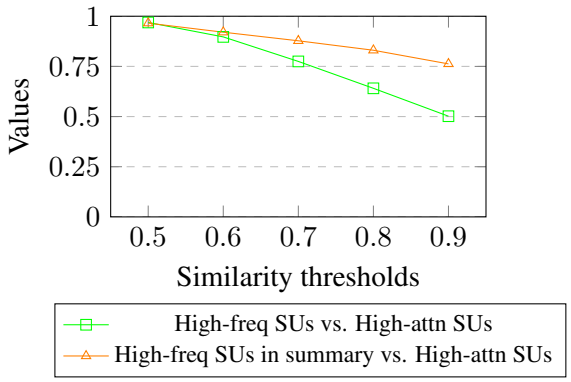
than those requiring massive pre-training. We will also discuss the situation where collecting training data with the lead bias characteristic is infeasible in our following experiments.

5.2 Can our model learn frequency through attention mechanism?

In this experiment, we first collect high-frequency semantic units as ground truths using a pre-trained Sentence-BERT (Reimers and Gurevych, 2019). The Sentence-BERT model (Reimers and Gurevych, 2019) encodes the source text spans divided by a sliding window, and we obtain the corresponding semantic unit embeddings. Then, we compute the frequency of semantic units by calculating the cosine similarity between each two semantic unit embeddings. If the similarity score is above a defined threshold, the two semantic units are considered semantically similar, and we add the frequency of the semantic units by one.



(a) Recall between top $N\%$ high-frequency semantic units (gold) and top $N\%$ high-attention semantic units (prediction). The similarity threshold is set as 0.5.



(b) Recall between top 40% high-frequency semantic units (gold) and top 40% high-attention semantic units (prediction) under different similarity thresholds.

Figure 3: Comparison of high-attention semantic units and high-frequency semantic units. Green line: comparison between high-frequency semantic units in source articles and high-attention semantic units in source articles; Orange line: comparison between high-frequency semantic units in summaries and high-attention semantic units in source articles.

We use recall to compare the overlapping rate between the top N % high-attention semantic units captured by our model and the top N % high-frequency semantic units decided by Sentence-BERT embeddings. The former is obtained by selecting the semantic units with top N % highest attention weights as mentioned in Section 3.3 and is considered our model predictions. The latter is referred to as ground truths. Figure 3a (the green line) shows that we can capture most of the high-frequency semantic units using the attention mechanism in our proposed method. Even when only the top 5% high-frequency semantic units are considered, we still successfully capture approximately 85% of the correct high-frequency semantic units.

We then inspect the performance under different similarity thresholds to see if two semantic units are also semantically similar given stricter conditions. In Figure 3b, the scores drop when the threshold is higher because semantic units are less likely to be matched. Nevertheless, the recall is 50% when the similarity threshold is 0.9, which means our model can retrieve approximately half of the correct high-frequency semantic units under harsh measurement conditions.

We also investigate the overlapping rate between the high-attention semantic units retrieved by our model and the high-frequency semantic units that also appear in the gold summaries. We use Sentence-BERT embeddings, as mentioned in this section before, to obtain the high-frequency semantic units in the gold summaries and show the results in Figure 3a (orange line). We find the trend is similar to that of comparing high-attention semantic units and the high-frequency semantic units presented only in the source articles (green line in Figure 3a). In Figure 3b, the recall remains high even if a higher similarity threshold is set. The results suggest that our model can capture most of the salient high-frequency semantic units that are also included in the gold summaries.

5.3 Generate summaries with high-frequency semantic units

This section investigates if we can use semantic units alone to generate extractive summaries. Here, we introduce two baseline methods for per-

Settings	R1	R2	RL
Extracting tokens from high-frequency SUs as summaries *(baseline)*	24.03	6.74	20.94
Extracting sentences with high-frequency SUs as summaries *(baseline)*	32.53	11.32	29.24
Extracting SUs in the sentences with high-attention SUs to decode *(current)*	37.54	14.49	33.52
Extracting SUs in the sentences with high-ROUGE SUs to decode *(optimal)*	41.12	18.13	37.08

Table 2: ROUGE F_1 scores on the *CNN/Daily Mail* dataset with different semantic unit selection methods when decoding twice.

formance comparisons. In Table 2, the first baseline simply extracts the corresponding tokens in the high-frequency semantic units computed by Sentence-BERT (Reimers and Gurevych, 2019) embeddings as the summaries. The second baseline further calculates the sentence score for each sentence by averaging the frequency of the semantic units in a sentence, where the frequency is also computed using Sentence-BERT embeddings. The sentences with the highest scores are concatenated into a summary of a maximum sequence length l. Thus, the second baseline can be viewed as extractive summarization using sentence-level frequency information. According to Table 2, our proposed abstractive method can obtain higher ROUGE scores than the two baselines, implying our method can effectively leverage semantic units for the summarization task.

5.4 Optimal performance with high-ROUGE semantic units

The last row of Table 2 presents the upper bound of our model performance. We directly take the semantic units included in the source sentences that maximize the ROUGE-2 score with respect to the gold summary, and the selected semantic units are the inputs for the second round of decoding. The results show that our model can generate better summaries if it puts more attention on the salient parts that are more likely to appear in the human-written summaries. In short, semantic unit selection is crucial for our model because it significantly affects the final performance.

5.5 Low-resource language

In Table 3, we present the performance of our model trained on the MLSUM (Scialom et al., 2020) dataset, which contains news articles in Russian, to check our model performance on data in low-resource language. It is noted that the MLSUM-RU news summaries have a higher level of abstractiveness than that of *CNN/Daily Mail*. In addition, the articles in the MLSUM-RU dataset have no lead bias characteristic, and the amount of data is far less than that of *CNN/Daily Mail*. The result shows that our model achieves a higher ROUGE-L[3] than that of the supervised pointer-generator network (See et al., 2017) and the lead-3 extractive baseline in the low-resource setting. The multilingual-BERT with 340M parameters, the largest model among the three, is pre-trained in a supervised manner and yields the best performance, as expected. The result also highlights that the scenario where there is difficulty collecting enough data for pre-training or collecting data using the lead bias characteristic does exist. Further experiments with different dataset sizes and transfer learning are also provided in Appendix A.6 and A.7.

Model	MLSUM-RU (len 15) (26k)
Lead-3	5.94
Pointer-generator	5.71
Multilingual-BERT	9.48
Ours	**6.87**

Table 3: ROUGE-L F_1 scores on the MLSUM Russian dataset. The desired length 15 for the summaries and the data size 26k are also appended in the table.

6 Conclusion

In this work, we propose an unsupervised abstractive summarization model using semantic units. The frequency of semantic units helps determine whether a specific content is more likely to be included in a human-generated summary. Our model learns to discriminate semantic units from the source articles by frequency through the proposed two-stage training and the inference workflow. The proposed model can achieve competitive ROUGE scores without paired data or pre-trained models compared to the large-scale pre-training methods and the methods under the same unsuper-

[3]Here we aggregate tokens within a sliding window by adding the beginning and the last token embeddings for constructing semantic unit embeddings. The ROUGE-L score with the averaging method (Avg.) is 6.08 for MLSUM-RU. See Appendix A.3 for more details.

vised settings. Our method is a potential solution for real-world scenarios where directly applying pre-trained models or collecting data with the lead bias characteristic is infeasible.

7 Acknowledgements

We would like to thank the reviewers for their insightful comments. We also thank Chien-Liang Liu and Szu-Tung Lin for reviewing our code. This work was funded in part by Qualcomm through a Taiwan University Research Collaboration Project NAT-487842 and in part by the Ministry of Science and Technology, Taiwan, under grant MOST 111-2221-E-006-001.

References

Christos Baziotis, Ion Androutsopoulos, Ioannis Konstas, and Alexandros Potamianos. 2019. SEQˆ3: Differentiable sequence-to-sequence-to-sequence autoencoder for unsupervised abstractive sentence compression. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 673–681, Minneapolis, Minnesota. Association for Computational Linguistics.

Arthur Bražinskas, Mirella Lapata, and Ivan Titov. 2020. Unsupervised opinion summarization as copycat-review generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5151–5169, Online. Association for Computational Linguistics.

Eric Chu and Peter Liu. 2019. Meansum: a neural model for unsupervised multi-document abstractive summarization. In *International Conference on Machine Learning*, pages 1223–1232. PMLR.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Shibhansh Dohare, Vivek Gupta, and Harish Karnick. 2018. Unsupervised semantic abstractive summarization. In *Proceedings of ACL 2018, Student Research Workshop*, pages 74–83, Melbourne, Australia. Association for Computational Linguistics.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 Text Summarization Workshop*, pages 1–8.

Alexander Fabbri, Simeng Han, Haoyuan Li, Haoran Li, Marjan Ghazvininejad, Shafiq Joty, Dragomir Radev, and Yashar Mehdad. 2021. Improving zero and few-shot abstractive summarization with intermediate fine-tuning and data augmentation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 704–717, Online. Association for Computational Linguistics.

Thibault Févry and Jason Phang. 2018. Unsupervised sentence compression using denoising auto-encoders. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 413–422, Brussels, Belgium. Association for Computational Linguistics.

Karl Moritz Hermann, Tomás Kočiskỳ, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, volume 28, pages 1693–1701. Curran Associates, Inc.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Masaru Isonuma, Junichiro Mori, and Ichiro Sakata. 2019. Unsupervised neural single-document summarization of reviews via learning latent discourse structure and its ranking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2142–2152, Florence, Italy. Association for Computational Linguistics.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Philippe Laban, Andrew Hsi, John Canny, and Marti A. Hearst. 2020. The summary loop: Learning to write abstractive summaries without examples. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5135–5150, Online. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

Peter J. Liu, Yu-An Chung, and Jie Ren. 2019. Summae: Zero-shot abstractive text summarization using length-agnostic auto-encoders. *ArXiv*, abs/1910.00998.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 145–152, Boston, Massachusetts, USA. Association for Computational Linguistics.

Ani Nenkova and Lucy Vanderwende. 2005. The impact of frequency on summarization. *Microsoft Research, Redmond, Washington, Tech. Rep. MSR-TR-2005*, 101.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2020. MLSUM: The multilingual summarization corpus. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8051–8067, Online. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Guokan Shang, Wensi Ding, Zekun Zhang, Antoine Tixier, Polykarpos Meladianos, Michalis Vazirgiannis, and Jean-Pierre Lorré. 2018. Unsupervised abstractive meeting summarization with multi-sentence compression and budgeted submodular maximization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 664–674, Melbourne, Australia. 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 generation. In *International Conference on Machine Learning*, pages 5926–5936. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 30:5998–6008.

Yaushian Wang and Hung-Yi Lee. 2018. Learning to encode text as human-readable summaries using generative adversarial networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4187–4195, Brussels, Belgium. Association for Computational Linguistics.

Ziyi Yang, Chenguang Zhu, Robert Gmyr, Michael Zeng, Xuedong Huang, and Eric Darve. 2020. TED: A pretrained unsupervised summarization model with theme modeling and denoising. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1865–1874, Online. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Jiawei Zhou and Alexander Rush. 2019. Simple unsupervised summarization by contextual matching. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5101–5106, Florence, Italy. Association for Computational Linguistics.

Chenguang Zhu, Ziyi Yang, Robert Gmyr, Michael Zeng, and Xuedong Huang. 2019. Make lead bias in your favor: Zero-shot abstractive news summarization. *arXiv preprint arXiv:1912.11602*.

A Appendix

A.1 Human evaluation

We present the human evaluation results on the linguistic qualities of the generated summaries using our model and that of Wang and Lee (2018). We follow the definitions and instructions for scoring on DUC 2007. We asked three workers on Mechanical Turk to score the five dimensions: grammaticality, non-redundancy, referential clarity, focus, and structure/coherence. We sampled 100 summaries in total, including 50 summaries generated by our model and the other 50 summaries generated by Wang and Lee's (2018). Table 4 shows that the qualities of our generated summaries are slightly better than those of Wang and Lee (2018) in most dimensions except for non-redundancy. This is probably because our model cannot differentiate similar content since our model only learns to discriminate semantic unit frequency.

A.2 Frequent content statistics

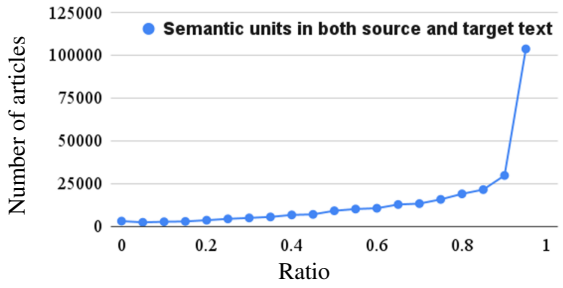


Figure 4: The figure compares high-frequency semantic units and semantic units in the summary of each article in *CNN/Daily Mail*, which includes 287k article-summary pairs in total. The x-axis represents the ratio of high-frequency semantic units which also show up in summaries. The y-axis is the number of articles in the *CNN/Daily Mail* training set. The threshold of the cosine similarity is set as 0.5.

Nenkova and Vanderwende (2005) proved that content unit frequency could help determine if a specific unit of content is more likely to appear in a human-written summary. We thus investigate if such a tendency also holds in recent summarization dataset, *CNN/Daily Mail* (Figure 4). We compute the frequency of the semantic units for each source article in the *CNN/Daily Mail* dataset as mentioned in Section 5.2; We can clearly observe that, in *CNN/Daily Mail*, about two third of the source articles in which over half of the high-frequency semantic units are included in a summary. It strongly supports our assumption that the frequency of semantic units in the source text can provide information that helps summarization.

A.3 Semantic unit construction

Since constructing semantic unit embeddings is similar to making span representations, we experiment with three span aggregation methods (Table 5). The first (Sum) is to add the beginning and last token embeddings within a semantic unit window. The second method (Cat.) is to concatenate the beginning and the last embeddings within a semantic unit. We note here that the second method requires an extra linear layer to adapt the concatenated representations into the defined input size of the decoder. The last method (Avg.) uses the averaged embeddings within a semantic unit as the final semantic unit embeddings. We adopt the last method to construct the semantic unit embeddings in our final model, as it does not require extra model parameters and yields the highest ROUGE scores among the three.

We tested different sliding window sizes c of 5, 7, and 9 when constructing semantic units. This range was determined considering two reasons: it was hard to form a basic meaning (e.g., a subject, an object, and a verb) with only three tokens where the BERT subword-level tokenizer (Devlin et al., 2019) was used in our experiments. Furthermore, there are 10.42 tokens, on average, in a clause in the *CNN/Daily Mail* training set. A larger sliding window size results in slightly fewer semantic unit embeddings for each article, and the number of semantic units sharing the same tokens also increases. The results are shown in Table 6. Among the three settings, the model with a window size of 5 yielded the highest ROUGE scores, and the performance gradually dropped when the sliding window size was larger. Therefore a sliding window size of 5 was adopted in our final model.

A.4 Effect of applying attention weights to decode again

The results presented in Table 7 prove that decoding twice leads to better performance than decoding once with unweighted semantic units. Thus, applying learned attention weights as masks for semantic units should help the model focus on salient information.

	Grammaticality	Non-redundancy	Referential clarity	Focus	Structure and Coherence
Unsupervised GAN	2.6	**3.3**	3.4	3.4	2.9
Ours	**3.0**	3.0	**3.8**	**3.9**	**3.4**

Table 4: Linguistic quality human evaluation scores (scale 1-5, higher is better).

Settings	R1	R2	RL
Sum	36.94	13.28	32.68
Cat.	34.16	10.80	30.30
Avg.	**37.54**	**14.49**	**33.52**

Table 5: ROUGE F_1 scores on *CNN/Daily Mail* test set with different aggregation methods for constructing semantic units.

Settings	R1	R2	RL
Window size 5	**36.94**	**13.28**	**32.68**
Window size 7	36.18	11.85	31.84
Window size 9	33.94	9.32	29.47

Table 6: ROUGE F_1 scores on *CNN/Daily Mail* test set with different sliding window sizes for constructing semantic units. We use Sum as the aggregation method for semantic unit embeddings.

A.5 Two-stage training

Our training process has two stages: masked semantic units prediction and reconstruction from semantic units, as introduced in Sections 3.2.1 and 3.2.2. We then attempt to determine experimentally if the two-stage training strategy helps summarization. Among the three settings in Table 8, the model with only the first-stage training obtains the worst performance, which shows that training the model to predict the words in the masked semantic units is inadequate for summarization purposes. Furthermore, training the model with the second stage brings much higher ROUGE scores than the "first stage only" setting. We infer that the reconstruction stage significantly affects the performance. Note that the number of training steps in Table 8 was greater than the one we mentioned in Section 4.1 to make the number of training steps in all the settings the same for the "first stage only" setting needs more training steps.

Decoding times	R1	R2	RL
1	34.26	11.71	30.27
2	**36.94**	**13.28**	**32.68**

Table 7: ROUGE F_1 scores on the *CNN/Daily Mail* test set with different decoding times using Sum as the aggregation method for semantic unit embeddings.

Settings	R1	R2	RL
2-stages *(current)*	**37.01**	13.27	**32.80**
First stage only	28.96	5.30	25.64
Second stage only	36.54	**14.00**	32.70

Table 8: ROUGE F_1 scores on the *CNN/Daily Mail* test set under various settings for the training stages. We use Sum as the aggregation method for semantic unit embeddings. Each setting is trained using the same number of steps.

A.6 Transfer learning

Settings	R1	R2	RL
Train on CNN/DM	**36.94**	**13.28**	**32.68**
Train on XSum	35.31	11.85	31.29
Train on Wikipedia	34.77	11.53	30.53

Table 9: ROUGE F_1 scores on the *CNN/Daily Mail* test set (target domain) when trained under different sources. We use Sum as the aggregation method for semantic unit embeddings.

We trained our model on other sources and tested it on the *CNN/Daily Mail* test set. The results are shown in Table 9. Since XSum is also an English news summarization dataset with a similar data size scale compared with *CNN/Daily Mail*, the performance difference was minimal, as expected. However, the performance was still comparable when the model was trained on Wikipedia, which is in a different domain from the news domain. This result shows that our model is capable of summarizing even if the source of the training data is different.

A.7 Data size

In the following experiment, we inspect our model performance on various training data sizes that range from 1k, 10k, and 100k to the complete 287k articles in *CNN/Daily Mail* to simulate the low-resource setting. The ROUGE scores are presented in Table 10. We can observe that even with only a third of the data, our model still yields comparable performance compared to the model trained with the complete data. Nevertheless, deep learn-

	Ours			Pointer-generator network (See et al., 2017)		
	R1	**R2**	**RL**	**R1**	**R2**	**RL**
Full data	36.94	13.28	32.68	39.53	17.28	36.38
100k	35.78 (-3.14%)	11.72 (-11.75%)	31.58 (-3.37%)	32.33 (-18.21%)	10.80 (-37.50%)	29.85 (-17.95%)
10k	26.69 (-27.75%)	4.47 (-66.34%)	23.15 (-29.16%)	28.11 (-28.89%)	7.40 (-57.18%)	25.75 (-29.22%)
1k	17.20 (-53.44%)	1.11 (-91.64%)	15.13 (-53.70%)	23.00 (-41.54%)	2.79 (-83.85%)	20.77 (-42.91%)

Table 10: ROUGE F_1 scores on different training data sizes for *CNN/Daily Mail*. The full data is 287k articles in total.

ing models still require a certain number of training data to tune the million-scaled model parameters. The performance drops significantly when the amount of data is decreased to one-tenth of the original data size. We also compare the effects of different training data sizes with a supervised system, the pointer-generator network (See et al., 2017). The results show that our model performance and the pointer-generator network both decrease when the data size is small. However, our model performance decreases less than the case for the supervised system with 100k training articles. Also, with only 10k training articles, our performance is comparable to the supervised system. However, when the amount of training data is significantly small, for example, 1k articles, supervised systems appear to yield better results than unsupervised systems.