A Preliminary Exploration of GANs for Keyphrase Generation

Avinash Swaminathan[1], Haimin Zhang[2], Debanjan Mahata[2], Rakesh Gosangi[2], Rajiv Ratn Shah[1], Amanda Stent[2]

[1] IIIT-Delhi, India, [2] Bloomberg, New York
s.avinash.it.17@nsit.net.in, {dmahata, hzhang449, rgosangi}@bloomberg.net, astent@bloomberg.net, rajivratn@iiitd.ac.in

Abstract

We introduce a new keyphrase generation approach using Generative Adversarial Networks (GANs). For a given document, the generator produces a sequence of keyphrases, and the discriminator distinguishes between human-curated and machine-generated keyphrases. We evaluated this approach on standard benchmark datasets. We observed that our model achieves state-of-the-art performance in the generation of abstractive keyphrases and is comparable to the best performing extractive techniques. Although we achieve promising results using GANs, they are not significantly better than the state-of-the-art generative models. To our knowledge, this is one of the first works that use GANs for keyphrase generation. We present a detailed analysis of our observations and expect that these findings would help other researchers to further study the use of GANs for the task of keyphrase generation.

1 Introduction

Keyphrases capture the most salient topics of a document and are often indexed in databases to help with search and information retrieval techniques. Researchers tag their scientific publications with high-quality keyphrases to ensure discoverability in scientific repositories. Automatic identification of keyphrases is of great interest to the scientific community as it helps to recommend relevant articles, suggest missing citations to authors, identify potential peer reviewers, and analyze research trends (Augenstein et al., 2017).

Keyphrases could either be *extractive* (part of the document) or *abstractive* (not part of the document). Some prior works have referred to them as present and absent keyphrases, respectively. Keyphrase generation is the process of predicting both extractive and abstractive keyphrases from a given document. Most of the previous works in keyphrase domain, including both supervised and unsupervised techniques, primarily focus on extractive keyphrases (Hasan and Ng, 2014; Mahata et al., 2018; Sahrawat et al., 2020). Recent studies Meng et al. (2017); Ye and Wang (2018); Chan et al. (2019) have started to develop generative approaches that produce both abstractive and extractive keyphrases from documents. Though these studies have shown some promise, the results suggest that there is great room for improvement.

Most of the supervised natural text generation approaches use Maximum Likelihood Estimation (MLE) objective (Lu et al., 2018). However, MLE techniques have often been observed not to be generating satisfactory text because of exposure bias (Lu et al., 2018). Generative approaches based on Reinforcement Learning (RL) or adversarial training have been shown to address some of the challenges of exposure bias. One such example is the success in the field of summary generation using GANs (Wang and Lee, 2018). Driven by these developments, we posit that, as with summarization, keyphrase generation can also benefit from the use of GANs. To pursue this hypothesis, in this paper, we propose a new GAN architecture for keyphrase generation where the generator produces a sequence of keyphrases from a given document, and the discriminator distinguishes between human-curated and machine-generated keyphrases (Section 2).

We introduce an adversarial training setup for the state-of-the-art generative approaches used in (Chan et al., 2019) (Section 3). We also propose a hierarchical attention-based discriminator architecture. Our empirical analysis shows that the generative models, in GAN setup, improves generation of abstractive keyphrases but do not show any significant improvements for extractive keyphrases.

As concluded by (Çano and Bojar, 2019), we

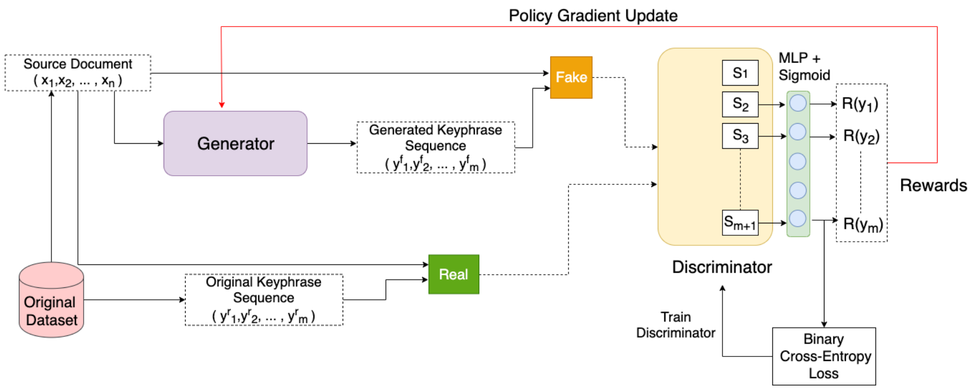
Policy Gradient Update
Source Document (x1,x2, ... , xn)
Generator
Generated Keyphrase Sequence (yf1,yf2, ... , yfm)
Fake
Real
Original Dataset
Original Keyphrase Sequence (yr1,yr2, ... , yrm)
S1
S2
S3
Sm+1
MLP + Sigmoid
R(y1)
R(y2)
R(ym)
Rewards
Discriminator
Train Discriminator
Binary Cross-Entropy Loss

Figure 1: GAN framework for generating keyphrases.

think keyphrase generation is still a hard text summarization challenge. We share our observations as well as our implementations with the scientific community. We expect these findings would help other researchers further explore the application of GANs for keyphrase generation. Following are our main contributions:

• We propose a GAN framework using reinforcement learning for keyphrase generation with a new discriminator architecture based on hierarchical attention that allows for providing reward to partially decoded sequences.

• We evaluate our proposed method on four publicly available datasets and compare our results with five state-of-the-art deep neural generative models for keyphrase generation[1].

2 Methodology

Given a document $T = \{x_1, x_2, ..., x_n\}$, where x_i is the i^{th} token, the problem of keyphrase generation is to generate a set of keyphrases $y = \{y_1, y_2, ..., y_m\}$ that best capture the semantic meaning of T. In this paper, we approach this as a supervised problem solved specifically using GAN (Goodfellow et al., 2014). Our GAN model consists of a generator G trained to produce a sequence of keyphrases from a given document, and a discriminator D that learns to distinguish between machine-generated and human-curated keyphrases.

Adversarial learning for text is a challenging task as it is not straight-forward to back-propagate the loss of the discriminator due to the discrete nature of text data (Rajeswar et al., 2017). We use RL to address this issue, where the generator is treated as an RL agent, and its rewards are obtained from the discriminator's outputs. Fig 1, shows an overview of our framework.

2.1 Generator

- We employ CatSeq (Yuan et al., 2018) as our generator. CatSeq model uses an encoder-decoder framework where the encoder is a bidirectional Gated Recurrent Unit (bi-GRU), and the decoder a forward GRU. For a given document, the generator produces a sequence of keyphrases: $y = \{y_1, y_2, ..., y_m\}$, where each keyphrase y_i is composed of tokens $y_i^1, y_i^2, ..., y_i^{l_i}$. To incorporate out-of-vocabulary tokens, we use a copying mechanism (Gu et al., 2016). We also use an attention mechanism to help the generator identify the relevant components of the source text.

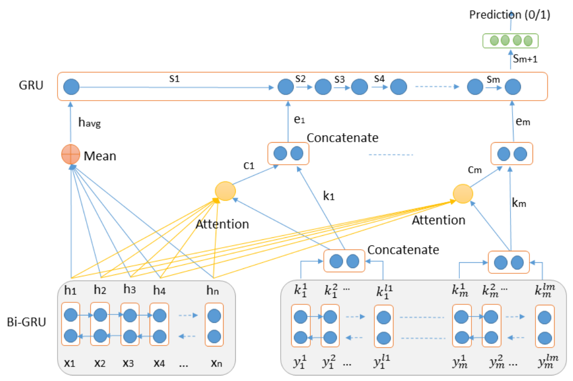

Figure 2: Schematic of Proposed Discriminator (D)

2.2 Discriminator

- The main aim of the discriminator is to distinguish between human-curated and machine-generated keyphrase sequences. To achieve this, the discriminator would also require a representation of the original document T. We proposed a new conditional hierarchical discriminator model (Fig 2) that

[1] Code - https://github.com/avinsit123/keyphrase-gan

consumes the original document T, a sequence of keyphrases y, and outputs that probability of the sequence being human-curated.

The first layer of this hierarchical model consists of $m + 1$ bi-GRUs. The first bi-GRU encodes the input document T as a sequence of vectors: $h = \{h_1, h_2, ..., h_n\}$. The other m bi-GRUs, which share the same weight parameters, encode each keyphrase y_j as a vector k_j, resulting in a sequence of vectors: $\{k_1, k_2, ..., k_m\}$. We then use an attention-based approach (Luong et al., 2015) to build context vectors c_j for each keyphrase (eq. 1), where c_j is a weighted average over h. By concatenating c_j and k_j, we get a contextualized representation $e_j = [c_j : k_j]$ of y_j.

$$c_j = \frac{\sum_{i=1}^{n} h_i \cdot e^{h_i w_s k_j}}{\sum_{i=1}^{n} e^{h_i w_s k_j}} \tag{1}$$

The second layer of the discriminator is a GRU which consumes the average of the document representations h_{avg} and all the contextualized keyphrase representations $e_1, e_2, ..., e_m$ as:

$$s_{t+1} = \begin{cases} GRU(h_{avg}, s_t), & if\ t = 0 \\ GRU(e_t, s_t), & otherwise \end{cases}. \tag{2}$$

The final state of this layer is passed through one fully connected layer (w_f) and sigmoid transformation to get the probability that a given keyphrase sequence is human-curated $P_h = \sigma(w_f s_{m+1})$.

2.3 Adversarial Training

The goal of the framework is to optimize the generator to produce keyphrase sequences that resemble human-curated keyphrase sequences. This is achieved by training the generator and discriminator in an alternating fashion. Namely, we train the first version of the generator using maximum likelihood estimation (MLE). We then use this generator to produce machine-generated keyphrases (S_f) for all documents. We combine them with the corresponding human curated keyphrases (S_r), and train the first version of the discriminator to optimize for the following loss function:

$$D_{loss} = -E_{y \in S_r}[log(D(y))] - E_{y \in S_f}[log(1 - D(y))] \tag{3}$$

To train the subsequent versions of the generator, we employ reinforcement learning, where the policy gradient is defined as:

$$\bigtriangledown R_G = \sum_{i=1}^{m} [D(y_i) - B] \bigtriangledown log \prod_{j=1}^{l_i} G(y_i^j | y_i^{1:j-1}, y_{1:i-1}, x) \tag{4}$$

B is a baseline obtained by greedy decoding of keyphrase sequence using self-critical sequence training (Rennie et al., 2016) method. The rewards for the generator are calculated from the outputs of the discriminator trained in the previous iteration. The resulting generator is then used to create new training samples for the discriminator. This process is continued until the generator converges.

When using RL for text generation (Li et al., 2017), it is necessary to support rewards for intermediate steps or partially decoded sequences. One of the advantages of our proposed discriminator architecture it can assign individual rewards to each generated keyphrase or one reward to the entire sequence. To support individual rewards, each state s_i of the final discriminator layer is passed through a feed-forward neural network with a sigmoid activation $R(y_i) = D(y_i) = \sigma(W_f s_{i+1})$. To obtain reward for the entire sequence, we just use the final predicted probability.

3 Experimental Work

3.1 Datasets

We trained the proposed GAN model using KP20k dataset (Meng et al., 2017) which consists of 549,818 samples (train: 509,818, test: 20,000, validation: 20,000). Each sample is a scientific article consisting of the title, abstract, and the corresponding human-assigned keyphrases. In addition to KP20k, we also evaluated the model on three other standard keyphrase datasets: Inspec (Hulth, 2003), NUS (Le et al., 2016), and Krapivin (Krapivin et al., 2009) which contain 500, 211, and 400 test samples respectively. We discarded all samples from KP20k training set that overlap with the other three test datasets.

3.2 Baselines

We compare our proposed GAN model against five state-of-the-art generative approaches: CatSeq, CatSeq-D, CatSeq-Corr, CatSeq-TG, CatSeq-RL as implemented in (Chan et al., 2019). All the baselines models were trained from scratch using MLE except for CatSeq-RL, which was trained using RL. We report the performance of our model under two different reward strategies: GAN_{MR}: one reward per keyphrase, and GAN$_{SR}$: one reward per keyphrase sequence.

Model	Inspec		NUS		Krapivin		KP20k	
	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$
CatSeq	0.236	0.286	0.336	0.399	0.269	0.360	0.291	0.367
CatSeq-D	0.214	0.264	0.321	0.393	0.268	0.352	0.285	0.363
CatSeq-Corr	0.240	0.292	0.315	0.384	0.271	0.352	0.289	0.365
CatSeq-TG	0.229	0.278	0.333	0.398	0.275	0.356	0.292	0.366
CatSeq-RL	0.250	**0.300**	**0.375**	**0.433**	0.287	0.362	**0.310**	**0.383**
GAN$_{SR}$	0.253	0.293	0.340	0.413	0.284	0.363	0.293	0.371
GAN$_{MR}$	**0.258**	0.299	0.348	0.417	**0.288**	**0.369**	0.303	0.378

Table 1: F1 scores for extractive keyphrases on 4 datasets

Model	Inspec		NUS		Krapivin		KP20k	
	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$	$F_1@5$	$F_1@M$
CatSeq	0.004	0.011	0.013	0.019	0.017	0.032	0.015	0.031
CatSeq-D	0.008	0.012	0.018	0.029	0.018	0.030	0.015	0.031
CatSeq-Corr	0.005	0.008	0.019	0.032	0.022	0.033	0.015	0.032
CatSeq-TG	0.005	0.007	0.025	**0.046**	0.023	0.037	0.017	0.033
CatSeq-RL	0.009	0.017	0.019	0.031	0.026	0.046	0.024	0.044
GAN$_{SR}$	0.011	0.013	0.016	0.034	0.032	0.051	0.028	0.039
GAN$_{MR}$	**0.013**	**0.019**	**0.026**	0.038	**0.042**	**0.057**	**0.032**	**0.045**

Table 2: F1 scores for abstractive keyphrases on 4 datasets

3.3 Experimental Settings

For the CatSeq generator model, the word embeddings were initialized to 300 dimensions and hidden layers to 150 units. For the hierarchical discriminator model, word embeddings were initialized to 200 dimensions, and the hidden layer set to 150 units. A dropout of 0.5 is applied to both discriminator layers. We pre-trained both the generator and discriminator using Adam optimizer at a learning rate of 0.001 and batch size of 64 and 32, respectively. Whenever the loss stops converging, we apply learning rate decay. During adversarial training, we switched to Adagrad optimizer with a learning rate of 0.00005. These parameter choices were driven by tuning on the KP20k validation dataset.

The generator model was first pre-trained for 3 epochs. The entire GAN architecture was then trained for four iterations. In each iteration, the discriminator was first trained using MLE for 4 epochs, then generator using policy gradient (with rewards from discriminator) for 4 epochs. We observed that GAN model started to diverge after the fourth iteration. The target sequence consisted of a semicolon-separated list of extractive keyphrases, followed by a tag, followed by another semicolon-separated list of abstractive keyphrases. The tag was useful in distinguishing between the extractive and abstractive keyphrases during evaluation.

3.4 Evaluation Metrics

We present the model performances in terms of F1@K (Yuan et al., 2018), and F1@M (M denotes the no. of unique keyphrases). F1@K is calculated by comparing the top K items ($y_{:K}$) of the generated keyphrase sequence with the ground-truth sequence. In our experimental work, we set $K = 5$. When calculating F1@M, we consider the entire sequence of generated keyphrases. For comparison between keyphrases, we first apply Porter stemming and then use exact matching (Chan et al., 2019).

3.5 Results

Table 1 summarizes the results in terms of F1@5 and F1@M for extractive keyphrases. Table 2 presents the same for abstractive keyphrases. For extractive keyphrases, GAN$_{MR}$ model obtains the best performance on Krapivin dataset and the second best performance on the other three datasets (NUS, Inspec, and KP20K). Overall, the CatSeq-RL seems to be doing best for extractive keyprhases with the GAN$_{MR}$ model being a very close second. In the case of abstractive keyphrases, GAN$_{MR}$ obtains the best performance across all datasets, with the exception of F1@M for NUS. Also, we observed that GAN$_{MR}$ consistently outperforms the GAN$_{SR}$ model. Since GAN$_{SR}$ assigns a single reward to an entire sequence, it is possible that bad keyphrases when present in good keyphrase sequence might receive a high reward. Likewise, a

Model	Extractive		Abstractive	
	F1@5	F1@M	F1@5	F1@M
CatSeq(Pre-trained)	0.291	0.367	0.015	0.031
GAN	0.292	0.365	0.014	0.032
GAN$_{GRU}$	0.299	0.374	0.022	0.038
GAN$_{MR}$	**0.303**	**0.378**	**0.032**	**0.045**

Table 3: F1 Scores for different discriminator structures on the KP20k data

Model	Inspec	Krapivin	NUS	KP20k
Catseq	0.87803	0.781	0.82118	0.804
Catseq-D	0.88232	0.772	0.8372	0.8242
Catseq-Corr	0.86242	0.781	0.8472	0.8312
Catseq-TG	0.87101	0.779	0.8214	0.8124
Catseq-RL	0.8602	**0.786**	0.83	0.809
GAN$_{MR}$	**0.891**	0.771	**0.853**	**0.85**

Table 4: α-nDCG@5 metrics

good keyphrase is present in a bad sequence might receive a low reward. This type of a reward scheme could prove detrimental to the training of the generator. On the other hand, GAN$_{MR}$ assigns reward to each keyphrase based on its quality and irrespective of the sequence.

3.6 Discriminator Structure

We conducted a small ablation study to understand how different aspects of the discriminator architecture contribute towards the performance. We compare our hierarchical discriminator against two simpler versions: GAN and GAN$_{GRU}$. In GAN, we remove the part of the architecture that conditions on the original document. In GAN$_{GRU}$, we remove the attention mechanism. Table 3 presents a comparison of these three architectures on KP20k dataset. We observe that both GAN$_{MR}$ model and GAN$_{GRU}$ outperform GAN suggesting that incorporating the original document is necessary for generation. Also, GAN$_{MR}$ outperforms GAN$_{GRU}$ showing that the attention mechanism was beneficial.

3.7 Keyphrase Diversity

We also evaluated the models in terms of α-nDCG@5 scores (Clarke et al., 2008). α-nDCG is an extension of the DCG ranking and is used to measure the diversity of content generated. It works by penalizing redundant keyphrases and rewarding new keyphrases. The results are summarized in Table 4. Our model obtains the best performance on three out of the four datasets, with the exception of CatSeq-RL performing the best on Krapivin. The GAN$_{MR}$ model shows a significant improvement in the α-nDCG@5 scores over the pre-trained CatSeq model, in case of KP20k GAN$_{MR}$ model improves by almost 6%.

4 Conclusions and Future work

In this paper, we proposed the first GAN architecture for keyphrase generation. The model consists of a generator that produces a sequence of keyphrases given a document, and a discriminator that distinguishes between human-curated and machine-generated keyphrases. The two components of the GAN model are trained in an alternating fashion: the scores from the discriminator are used in the policy update of the generator, the keyphrases produced by the generator are used in the training of the discriminator. Our results show that the proposed model obtains better performance in generating abstractive keyphrases but fails to outperform baseline model for extractive keyphrases.

Further analysis of the results (details in supplementary materials) suggest that our model is effective in removing duplicates and generating diverse keyphrases. However, we observed the recall of our model is capped by the CatSeq generator, it did not produce any new keyphrases. We also observed that that our model suffers with some of the common challenges observed in GAN training such as vanishing gradient and mode collapse. Recent works (Lu et al., 2018; d'Autume et al., 2019) have proposed some solutions to address these challenges and we plan to explore them. Future direction also include alternate architectures, reward schemes, and evaluation using human judges.

References

Martín Arjovsky and Léon Bottou. 2017. Towards principled methods for training generative adversarial networks. *ArXiv*, abs/1701.04862.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. Semeval 2017 task 10: Scienceie-extracting keyphrases and relations from scientific publications. *arXiv preprint arXiv:1704.02853*.

Erion Çano and Ondřej Bojar. 2019. Keyphrase generation: A text summarization struggle. *arXiv preprint arXiv:1904.00110*.

Hou Pong Chan, Wang Chen, Lu Wang, and Irwin King. 2019. Neural keyphrase generation via reinforcement learning with adaptive rewards. In *ACL*.

Charles Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. 2008. Novelty and diversity in information retrieval evaluation. *Proc. of the 31st ACM SIGIR*, pages 659–666.

Cyprien de Masson d'Autume, Mihaela Rosca, Jack Rae, and Shakir Mohamed. 2019. Training language gans from scratch. *arXiv preprint arXiv:1905.09922*.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial networks. *ArXiv*, abs/1406.2661.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1262–1273.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP '03, pages 216–223, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mikalai Krapivin, Aliaksandr Autaeu, and Maurizio Marchese. 2009. Large dataset for keyphrases extraction. Technical report, University of Trento.

Tho Thi Ngoc Le, Minh Le Nguyen, and Akira Shimazu. 2016. Unsupervised keyphrase extraction: Introducing new kinds of words to keyphrases. In *Australasian Conference on Artificial Intelligence*.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

Sidi Lu, Yaoming Zhu, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Neural text generation: Past, present and beyond. *arXiv preprint arXiv:1803.07133*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Debanjan Mahata, John Kuriakose, Rajiv Shah, and Roger Zimmermann. 2018. Key2vec: Automatic ranked keyphrase extraction from scientific articles using phrase embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 634–639.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. *arXiv preprint arXiv:1704.06879*.

Sai Rajeswar, Sandeep Subramanian, Francis Dutil, Christopher Pal, and Aaron Courville. 2017. Adversarial generation of natural language. *arXiv preprint arXiv:1705.10929*.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2016. Self-critical sequence training for image captioning. *CoRR*, abs/1612.00563.

Dhruva Sahrawat, Debanjan Mahata, Haimin Zhang, Mayank Kulkarni, Agniv Sharma, Rakesh Gosangi, Amanda Stent, Yaman Kumar, Rajiv Ratn Shah, and Roger Zimmermann. 2020. Keyphrase extraction as sequence labeling using contextualized embeddings. In *European Conference on Information Retrieval*, pages 328–335. Springer.

Yau-Shian Wang and Hung-yi Lee. 2018. Learning to encode text as human-readable summaries using generative adversarial networks. *CoRR*, abs/1810.02851.

Hai Ye and Lu Wang. 2018. Semi-supervised learning for neural keyphrase generation. *arXiv preprint arXiv:1808.06773*.

Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Daqing He, and Adam Trischler. 2018. Generating diverse numbers of diverse keyphrases. *ArXiv*, abs/1810.05241.

A Appendix

We further analyse the performance of the GAN model . All graphs indicate metrics for GAN models with multiple rewards trained on the KP20k train dataset, validated on KP20k validation dataset and tested on the Krapivin and Inspec dataset. The GAN$_{MR}$ model is trained for 4 *iterations* with each iteration being composed of 4 *epochs* of discriminator training followed by generator training.

A.1 Vanishing Gradients

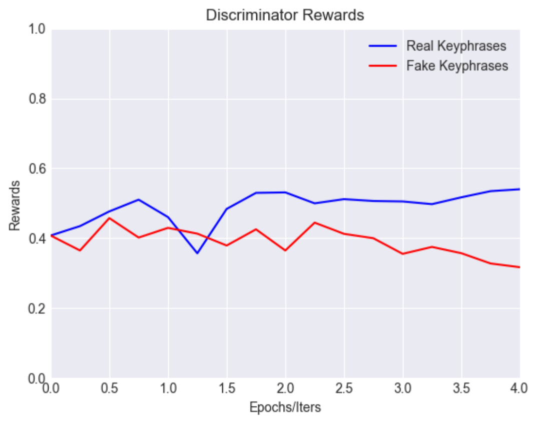

Figure 3: Discriminator Rewards while training GAN$_{MR}$ [2]

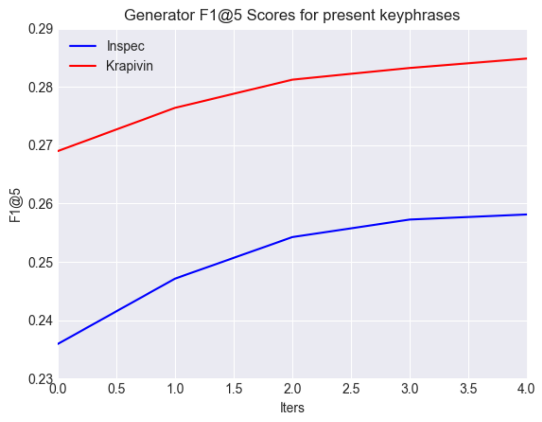

Figure 4: Generator's F1@5 Scores for present keyphrases

One problem we noticed in the GAN training process is as the discriminator performance improves, the ability of the generator to converge decreases. Fig. 3 shows after couple of iterations the discriminator becomes stronger, and assigns high rewards to all real keyphrases and low rewards to all

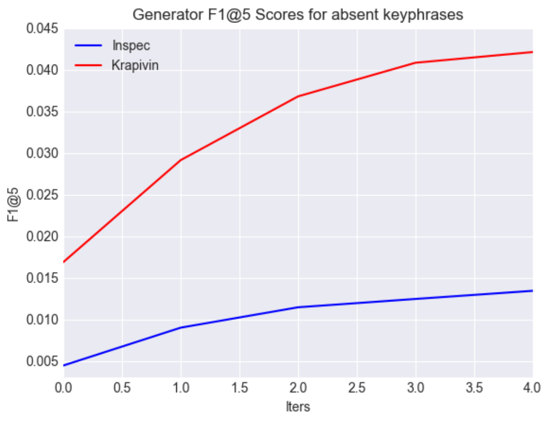

Figure 5: Generator's F1@5 Scores for absent keyphrases

fake keyphrases. While the magnitude of improvement in generator's F1 scores decreases for both present keyphrases (Fig 4) and absent keyphrases (Fig 5), and further training over the 4th iteration causes the generator to diverge.

This problem in GAN training is commonly known as *vanishing gradient problem* (Arjovsky and Bottou, 2017), where in the presence of stronger discriminator causes smaller gradients assigned to the generator during training. Thus, as the discriminator improves its strength throughout the iterations, it decreases the ability of the generator to converge.

A.2 Learning Rate

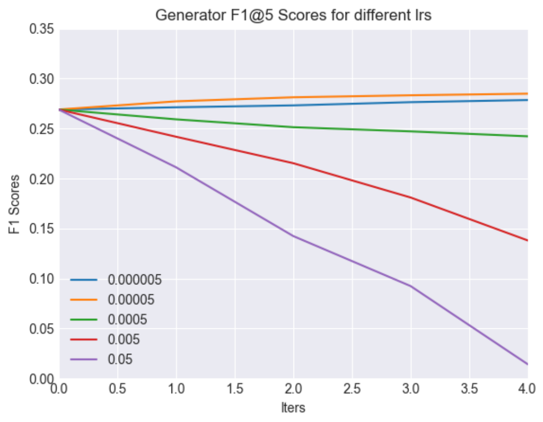

Figure 6: Changes in F1@5 scores of present keyphrases from Krapivin dataset for different learning rates

Some research shows increasing the learning rate might help the generator converge faster and overcome the gradient problem. Increase in the learning rate during GAN training, however causes the generator to enter *mode collapse*. In this case while the generator receives higher rewards from the discriminator, its performance with respect to F1 scores drastically decreases. Figure 6, shows the change in F1@5 scores for present keyphrases over the Krapivin dataset as we vary the learning rate. We observe that increasing the learning rate even by small amount leads to decrease in the F1 scores of the generator, and the decrease is more prominent as we increase the learning rate further, at learning rate 0.05, the generator starts generating gibberish text after 4 iterations.

A.3 Discriminator early stopping

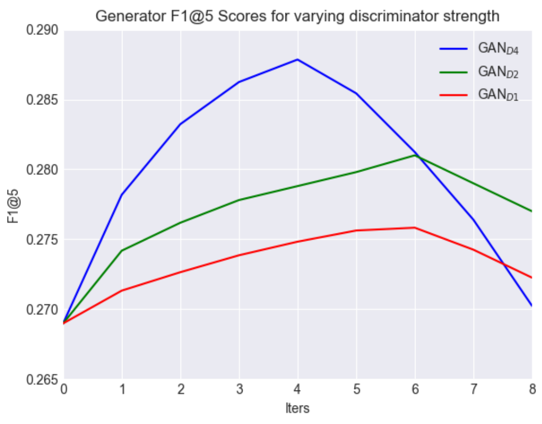

Figure 7: Change in F1@5 Scores on Krapivin dataset for varying strength of discriminator

The improvement in the strength of discriminator impedes the convergence of discriminator after each iteration of training. Thus, we experiment the effect of weaker discriminator on GAN training. The strength of the discriminator is reduced by early stopping and training it just for 1 and 2 epochs in each iteration instead of original 4 epochs . Table 5, shows performance for various generators on Krapivin dataset based on the strength of the discriminator. GAN$_{D1}$, GAN$_{D2}$ and GAN$_{D4}$ are the GAN models in which the discriminator is trained for 1, 2 and 4 epochs in each adversarial training iteration respectively. The F1 scores of these models indicate slowing down discriminator training actually end up with worse performance

Further analysis of an increase in generator's F1@5 scores on present keyphrases in fig 7 show that while training with a weak discriminator does

Model	Present		Absent	
	F1@5	F1@M	F1@5	F1@M
CatSeq(Pre-trained)	0.269	0.360	0.017	0.032
GAN$_{D1}$	0.276	0.364	0.028	0.039
GAN$_{D2}$	0.281	0.367	0.036	0.047
GAN$_{D4}$	**0.288**	**0.369**	**0.042**	**0.057**

Table 5: Generator's F1 scores on krapivin dataset for varying strength of discriminator

improve the generator's convergence in later iterations as evidence by GAN$_{D1}$ and GAN$_{D2}$ improving their F1 scores till 6th iterations, this magnitude of increase in not large enough. Fig 7 shows that initial increase in F1 scores of GAN$_{D2}$ in the 1st iteration is quite large compared to GAN$_{D2}$ and GAN$_{D1}$. This small improvement can be attributed to the fact that a weak discriminator find its difficult to distinguish between false and real keyphrases, thus assigning rewards closer to the baseline which do not reflect the reality of generated keyphrases causing the generator to converge less. We further see a manifestation of *vanishing gradient problem* in all 3 GAN models - the improvement in F1 scores decreases as we move across increasing iterations thus, this initial large increase is crucial in improving GAN$_{D4}$'s scores and making it the best performing model.

A.4 Examples

Table 6 and Tabel 7 shows examples of keyphrases generated by GAN$_{MR}$ and all baselines along with the original and author assigned keyphrases. Table 6 showcases some scenarios where the GAN$_{MR}$ improves upon existing keyphrases and Table 7 highlights examples where GAN$_{MR}$ performs worse.The following observations are made with respect to these examples.

1. **GAN$_{MR}$ model improves upon existing keyphrases generated by the pre-trained CatSeq model** - As seen in both examples of Table 6 the GAN model improves upon existing keyphrases generated by the pre-trained CatSeq model. In Ex.1, CatSeq generates keyphrases *object* and *image*. These differ from the original keyphrases *object recognition* and *image registration* only by 1 word. GAN$_{MR}$ model helps in introducing these new words and improves upon CatSeq-generated keyphrases by providing them with a higher reward and encouraging the generator. Similarly, in Ex.2 GAN$_{MR}$ improves

Source Abstract	*Affine invariants of convex polygons.* Registration and object recognition, proposed recently by yang and cohen are algebraically dependent. We show how to select an independent and complete set of the invariants. The use of this new set leads to a significant reduction of the computing complexity without decreasing the discrimination power.
CatSeq	affine invariants; convex polygons; registration; object ; object recognition; object; image
CatSeqTG	affine invariants; convex polygons;registration; object ;object recognition;pattern recognition;.
CatSeqCorr	affine invariants;convex polygons;registration; object; object recognition;
CatSeqD	affine invariants;convex polygons;registration; object;object recognition;image recognition;
GAN$_{MR}$	affine invariants; convex polygons; object; object recognition; image registration;
Original Keyphrases	affine invariants;convex polygons;object recognition;image registration;feature vector;convex quadruplet;complexity reduction;
Source Abstract	*The role of speech input in wearable computing.* Computers, and as we saw in this magazine's first issue, several companies are promoting products that use limited speech interfaces for specific tasks. However, we must overcome several challenges to using speech recognition in more general contexts, and interface designers must be wary of applying the technology to situations where speech is inappropriate.
CatSeq	*speech input;speech; speech;wearable computing;speech recognition;noise;*
CatSeqTG	*speech input; speech; speech; speech recognition;noise;.*
CatSeqCorr	*speech input; speech; speech; wearable computing;speech recognition;hidden markov models;*
CatSeqD	*speech input; speech; speech; wearable computing;speech recognition;*
GAN$_{MR}$	*speech input; wearable computing;speech recognition;background noise;*
Original Keyphrases	*speech input; speech interfaces; speech recognition; wearable computing;wearable computer;mobile speech recognition;background noise;speech recognizers;*

Table 6: Positive Examples generated by GAN$_{MR}$ and other baselines

Source Abstract	*Interestingness of frequent itemsets using bayesian networks as background knowledge.* The paper presents a method for pruning frequent itemsets based on background knowledge represented by a bayesian network. The interestingness of an itemset is defined as the absolute difference between its support estimated from data and from the bayesian network. Efficient algorithms are presented for finding interestingness of a collection of frequent itemsets and for finding all attribute sets with a given minimum interestingness. Practical usefulness of the algorithms and their efficiency have been verified experimentally. Categories and subject descriptors h.
CatSeq	*interestingness; frequent itemsets; bayesian networks; data mining;*
CatSeqTG	*interestingness; frequent itemsets; bayesian networks; data mining.*
CatSeqCorr	*interestingness; frequent itemsets; bayesian networks; background knowledge; data mining*
CatSeqD	*interestingness;frequent itemsets;bayesian networks;background knowledge;*
GAN$_{MR}$	*frequent items; bayesian networks;*
Original Keyphrases	*interestingness;frequent itemset;frequent itemsets;bayesian network;background knowledge;data mining;emerging pattern;association rule;association rules;*
Source Abstract	*Twenty years of the literature on acquiring out of print materials .* Out of print materials to assess recurring issues and identify changing practices. The out of print literature is uniform in its assertion that libraries need to acquire o.p.materials to replace worn or damaged copies, to replace missing copies, to duplicate copies of heavily used materials, to fill gaps in collections, to strengthen weak collections, to continue to develop strong collections, and to provide materials for new courses, new programs, and even entire new libraries.
CatSeq	*out of print; libraries; information retrieval;*
CatSeqTG	*out of print;*
CatSeqCorr	*out of print; libraries; information retrieval; data mining;*
CatSeqD	*out of print; print; libraries; united kingdom;*
GAN$_{MR}$	*out of print; retrieval;*
Original Keyphrases	*out of print materials; recurring issues; changing practices; library materials; out of print books; acquisition;*

Table 7: Negative Examples generated by GAN$_{MR}$ and other baselines

upon CatSeq-generated *Noise* by generating *Background Noise.*

2. **GAN$_{MR}$ model removes repeated and unwanted keyphrases and improves diversity**.
   All baseline models in Ex 2 of Table 6 generate keyphrase *speech* 2 times. However, GAN$_{MR}$ removes all repeating occurences of keyphrases and generates a diverse keyphrase sequence as evidenced by α-nDCG@5 metrics of diversity in Table 4.

3. **GAN$_{MR}$ model doesn't help the generator introduce new keyphrases**
   A consistent feature noticeable across all examples is that while the GAN model does improve upon generated keyphrases, it doesn't generate full new keyphrases. In none of the examples, does the GAN$_{MR}$ model introduce new keyphrases.

4. **GAN$_{MR}$ removes original keyphrases predicted by the pre-trained CatSeq model**
   Sometimes when the keyphrase is present in both the real and fake keyphrase sequence, the discriminator assigns low rewards to these keyphrases even though they might be true. These keyphrases often get discarded due to these low rewards during the GAN training process. Consider Ex.1 of Table 7, even though keyphrases *interestingness* and *data*

mining, both, are predicted by the CatSeq generator and are present in the original keyphrase sequence. However, the GAN$_{MR}$ model removes both these keyphrases generating a less original keyphrase sequence, thereby decreasing F1 score.

Thus GAN$_{MR}$ improves upon the CatSeq-generated keyphrases and removes repeated keyphrases. However, it falls short in generating new keyphrases thus not increasing the F1 score much.

A.5 Absent vs Present Keyphrases

GAN's cause an improvement in both present and absent keyphrases. While the improvement in F1 scores is more significant for present keyphrases, the marginal improvement in F1 scores of absent keyphrases is enough to make the GAN model perform better than all baselines. This improvement in F1 scores can be attributed to our reward scheme and the structure of the discriminator which gives equal priority in rewarding both absent and present keyphrases.

A.6 Conclusion

Analysis indicates that GAN training suffers from *vanishing gradient problem* preventing the generator from achieving maximum potential. Increasing the learning rate causes *mode collapse* and using a weaker discriminator does not guarantee a large increase in F1 scores. These observations indicate the inherent difficulty in training GANs on textual data especially keyphrases. The GAN model improves upon the MLE-Generated keyphrases and improves diversity by removing repeated keyphrases, however it fails to introduce new keyphrases.