Keyphrase Generation on News Articles

Eric Zhou, Mengxi Sun

Abstract

The generation of keyphrases that summarize an article is an important task for document retrieval, classification, and understanding. In this paper, we aim to improve keyphrase generation in the news domain using transformer-based language models. To achieve this goal, we improve upon former transformer models in three ways. First, we replace the copy mechanism and word-level tokenization of former models with subword-level tokenization in order to generate named entities and rare words. Second, we utilize pre-trained token embeddings from a large BART model fine-tuned on CNN Daily News articles for the summarization task. Third, we double the number of layers, the number of attention heads, and the size of the input token embeddings. As a result, we achieve a 7% improvement on the extraction of present keyphrases and a 12% improvement on the generation of absent keyphrases over the baseline transformer architecture.

Keyphrases: keyword generation, keyphrase generation, news summarization, sequence to sequence models, BART, NLP

1 Introduction

Keyphrases are short sequences of words that summarize the content of an article. They help search engines and human readers to quickly understand the content of an article before reading [13]. The automatic generation of keyphrases is a technology of immense value and broad application to many fields that need to classify, organize, or understand documents [14].

In this paper, we treat the keyphrase generation problem as a sequence to sequence modeling problem. Given a sequence of word tokens, the goal is to output a sequence of smaller, variable-length strings which summarize the content of the source document. Since the order of the sequence does not matter, and the sequence itself can contain a variable number of phrases, keyphrase generation is a set generation problem.

Prior neural networks have focused on generating keyphrases from the abstracts of scientific research articles [3, 11, 14, 22]. In this paper, we focus on the problem of generating keyphrases from news articles. The news article domain can be more challenging for several reasons. There are many more out-of-vocabulary named entities. The articles are hundreds of words longer [5]. Many articles include dialogue and a narrative voice, and thus sentences which are remotely related to the ground truth keyphrases. One of the goals of this paper is to take existing neural keyphrase generation networks and to characterize their strengths and weaknesses in the news domain.

The field of keyphrase generation has evolved quickly, from unsupervised topic modeling [1] to current transformer-based language models [13]. Recent research has shown that language models not only outperform unsupervised topic modeling methods for generating canonical present keyphrases. They also perform a task that topic modeling methods are unable to produce, which is the generation of relevant keyphrases that are absent in the source document [14].

In this paper, we reproduce the results from a recent state-of-the-art transformer model published in the last year, and we expand the transformer to achieve metrics that are 2-12% higher than what is reported in the literature. We describe in detail the training methodology used to achieve these metrics, and we diagnose both what the model does well and areas for improvement.

2 Related Works

We think about the task from the perspectives of topic analysis and absent keyphrase generation.

Present Keyphrase Extraction: Earlier works in topic extraction evaluate each document as a mixture model with the mixture components as topics and the number of underlying topics is known and fixed. In 2003, Blei, Ng and Jordan proposed the milestone method *latent Dirchlet allocation (LDA)*, which is a three-level generative probabilistic model [1]. Although LDA is powerful in extracting information groups, it requires sophisticated inference algorithms adjusting to any variant of the model. Later, neural topic models try to retain the performance of LDA while bypassing the complication in inference algorithms. Incorporating *Variational Auto-Encoder (VAE)*, *Neural Variational Document Model (NVDM)* published in 2016 achieves low perplexities [15] but produces incoherent topics in terms of word groups comprised of a topic and topic groups contained in a document by human judgement [20] [2]. The competing model — ProdLDA — changes assumption of words distribution from mixture model used in LDA to product of experts and uses the logistic normal distribution to explicitly approximate the Dirichlet distribution as prior for topic generation, improved the quality of generated topics from NVDM [17]. The most recent approach by Wang, Zhou, and He (2019) utilizing the *generative adversarial nets (GAN)* [6] and VAE outperforms all previous neural topic model in producing coherent topics [20]

Absent Keyphrase Generation: While the topic models discover latent patterns of documents in the corpus, the patterns are represented by groups of keywords present in the source. The sequence to sequence models allow machine to learn semantic meaning and generate keyphrases absent from the source. The first paper to adapt the architecture is "**CopyRNN**" in 2017, which generates one keyphrase per document using a recurrent neural network (**RNN**) architecture with a **copy-mechanism** to predict rare out-of-vocabulary words. **CatSeqD** is the first model to shift from generating single keyphrases to generating a variable-length sequence of keyphrases [22]. The model generates a ***seq****uence* of keyphrases *con****cat****enated* by delimiter tokens (`<sep>`), and it employs additional regularization techniques to ensure that the generated keywords are ***d****iverse*. There have been many other contributions since then. CorrRNN introduced additional regularization terms to reduce the correlation between generated keyphrases and keyphrase duplication [3]. Another team proposed semi-supervised methods to take advantage of unlabeled text [21]. And in the following year, an architecture that took into account structure information (such as the title of the source text) appeared [4]. Later, researchers proposed creating separate models for the present keyphrase extraction task and absent keyphrase generation task [12].

3 Datasets

Extensive research on keyphrase generation focuses on the academic article domain [3, 11, 14, 22]. We are interested in applying the *supervised keyphrase generation* techniques to the news domain, mainly KPTimes [5]. In the paper that releaed KPTimes, researchers showed that existing models, including state-of-the-art **CopyRNN**, do not generalize well across domains [5]. Here we discuss reasons why neural language models do not generalize well between the science paper and news domains.

The advantage of both the KPTimes and KP20k datasets over previous datasets in the respective domains is the much larger size sufficiently for neural-based training. Moreover, both of the datasets are particularly suited for absent keyphrases generation because both datasets annotate higher percentage of the keyphrases absent from the text than other datasets in their respective domains. Table 1 presents the details of the datasets.

The texts in KP20k use a lot of scientific terminologies created by the researchers with logic or academic convention behind, which are usually quite difficult to capture merely from the patterns of vocabulary or even semantic meanings. It is one of the main difficulties in generating absent keyphrases for scholar datasets. However, the news domain data contains a wider range of topics, named entities, people names, places, and their abbreviations, which appeared in the articles infrequently. Some of the concepts are important but others are not. This posts a major challenge in learning on a news dataset, making absent keyphrases generation for a news dataset even harder than for a single field scholar dataset.

Domain	Dataset	Ann.	#Train	#Dev	#Test	#words	#kp	len. kp	%abs
scholar	**KP20k**	A	530K	20K	20K	176	5.3	2.6	42.6
news	**KPTimes**	E	260K	10K	10K	921	5.0	1.5	54.7
scholar	SemEval	A&R	144	-	100	7961	14.7	2.2	19.7
news	DUC	R	-	-	308	847	8.1	2.0	3.7

Notes: We only train and test the models on KP20k and KPTimes. The other two datasets SemEval-2010 [8] and DUC-2001 [19] are included here for dataset comparison in the news and scholar domains. The statistics in this table are from [5]. The ground-truth annotation is performed by authors (A), readers (R) or editors (E). The number of documents in the training (#Train), validation (#Dev) and testing (#Test) splits are shown. The average number of keyphrases (#kp) and words (#words) per document, the average length of keyphrases (len. kp) and the ratio of keyphrases in the reference absent in the document (%abs) are computed on the test set.

Table 1: Dataset Comparison

In addition to the domain difference, there are others features that differ between KPTimes and KP20k, which may affect the performance of the models on these datasets. First, the keyphrases of KPTimes are curated by professional editors who follow a systematic procedures of keyword-tagging and are aided by automatic tag suggestions, as opposed to much less systematic author annotated KP20k. This may make the KPTimes training relatively easier because there is less variation in expert-annotated keyphrases. The second difference is that the text sources of KP20k are the abstracts of research papers, which are already the summarized version and om average around 750 words shorter than the whole news articles in the KPTimes.

4 Evaluation Methods

In this section, we define the metrics we use to evaluate our model, in particular, $\mathbf{F_1@10}$, $\mathbf{F_1}@\mathcal{O}$, and $\mathbf{R@50}$. These are respectively the $\mathbf{F_1}$ score for the top 10 keyphrases generated by the model, the $\mathbf{F_1}$ score for the top k keyphrases where k is the number of ground truth keyphrases, and the recall for the top 50 predicted keyphrases. Below, we explain the rationale for choosing these metrics.

The $\mathbf{F_1}$ metric is a combination of the $precision$ metric and the $recall$ metric, originating from the information retrieval field. Precision is the proportion of generated keywords which are present in the ground-truth. Recall is the proportion of ground truth keywords that were generated by the model. This is a measure of how comprehensive the model is, or how many ground-truth keyphrases were "recalled" by the model. $\mathbf{F_1}$ is the harmonic mean of precision and recall.

Formally, let the list of ground truth keyphrases be $\mathcal{Y}$ and the list of generated keyphrases be $\hat{\mathcal{Y}} = (\hat{y}_1, ..., \hat{y}_m)$. Let the top k predictions be $\hat{\mathcal{Y}}_{:k} = (\hat{y}_1, ..., \hat{y}_{\min(k,m)})$. Then, precision $\mathbf{P}$, recall $\mathbf{R}$, and $\mathbf{F_1}$ of the top k keyphrases are defined as follows:

$$\mathbf{P}@k = \frac{\hat{\mathcal{Y}}_{:k} \bigcap \mathcal{Y}}{\hat{\mathcal{Y}}_{:k}}$$
$$\mathbf{R}@k = \frac{\hat{\mathcal{Y}}_{:k} \bigcap \mathcal{Y}}{\mathcal{Y}}$$
$$\mathbf{F}_1@k = \frac{2 * \mathbf{P}@k * \mathbf{R}@k}{\mathbf{R}@k + \mathbf{P}@k}$$

Common metrics in this field are $\mathbf{F_1@5}$, $\mathbf{F_1@10}$ for evaluating the top 5 and 10 generated keyphrases respectively [14]. Others include $\mathbf{F_1}@\mathcal{O}$ where $k = |\mathcal{Y}|$, the number of ground truth keyphrases (and $\mathcal{O}$ stands for "oracle" for ground-truth). Less commonly, there is $\mathbf{F_1}@\mathcal{M}$ where $k = |\hat{\mathcal{Y}}|$, the number of generated keyphrases [13]. For evaluating absent keyphrases, $\mathbf{R@50}$ is common [13, 22]; we want to see if the model can generate any absent keyphrases in any of its top 50 predicted keyphrases.

5 Model

For this project, we implement two transformer-based models, the smaller of which, CatSeqD-Transformer, serves as our baseline, the larger of which, CatSeqD-BART-CNN, serves as our model extension. CatSeqD-BART-CNN, in addition to being larger, uses subword tokens and pretrained embeddings. First, we discuss our baseline model and how we reproduce published results on the KP20k dataset. Then we discuss how we train both the baseline model and the extended model on KPTimes.

Attribute	**Baseline** CatSeqD-Transformer	**Extension** CatSeqD-BART-CNN
Number of Encoder Layers	6	12
Number of Decoder Layers	6	12
Number of Attention Heads	8	16
Word or Subword Tokenization	Word (spaCy)	Subword (RoBERTa)
Token Embedding Dimensions	512	1024
Pretrained Token Embeddings?	No	Yes
Copy Mechanism Used?	Yes	No
Number of Steps Trained (on KPTimes)	70000	80000
Number of Hours Trained (on KPTimes)	60	100
Trained on KP20k in Prior Literature?	Yes	No
Trained on KPTimes in Prior Literature?	No	No
Code Base	OpenNMT [9]	Fairseq [16]

Table 2: Comparison between the baseline model and our extension

5.1 Baseline Model

5.1.1 Baseline Model Selection

We chose to re-implement the CatSeqD-Transformer model for this proposal, based on a survey of model architectures in "An Empirical Study on Neural Keyphrase Generation" by Meng et al. [13]. While the performances of all of the architectures were comparable, not differing by more than 2 percentage points on the $\mathbf{F_1}$ metrics, we selected the CatSeqD-transformer architecture for two main reasons:

1. It determines for itself when to stop generating new keyphrases, as opposed to other models which select the top k keyphrases where k is pre-determined.
2. It marginally outperforms the other models in extracting present keywords by about one percentage point.[1]

5.1.2 Baseline Model Reimplementation

Our code for model re-implementation can be found in this GitHub repository: https://github.com/eric-zhizu/OpenNMT-kpg-release

It is based on the code released by the review paper described above [13], which in turn is based on the OpenNMT (neural machine translation) python library [9].

The architecture of the model, which is implemented by the open-source OpenNMT library, is essentially the same as the transformer architecture found in "Attention is All You Need" [18].

Notably, despite the fact that we call this model "CatSeqD-Transformer", it does not employ the two regularization techniques found in the original CatSeqD paper [22]. The original CatSeqD is GRU-based and uses orthogonal regularization and semantic coverage to combat hidden state similarity at

[1] For a more extensive discussion of the models covered in this review, see our midterm report.

delimiter tokens (<sep>); however, our transformer-based implementation does not encounter the same issue.[2]

5.1.3 Baseline Training Procedure on KP20k

Present ($\mathbf{F_1@\mathcal{O}}$)		Present ($\mathbf{F_1@10}$)		Absent ($\mathbf{R@50}$)	
Ours	**Original**	**Ours**	**Original**	**Ours**	**Original**
36.1	36.2	28.9	29.0	11.1	15.0

Table 3: Reproducing the Reported Results on KP20k - Comparing our CatSeqD-Transformer trained with 100,000 steps (Ours) with the original paper's CatSeqD-Transformer trained with 300,000 steps (Original) on the KP20k dataset.

Training was performed on the KP20k scientific paper dataset as it was performed in the original paper [14]. The 50,000 most frequent words were set as the model's vocabulary. Word vector embeddings were of 512 dimensions and randomly initialized. Prior to inputting the data, the ground truth keyphrases were sorted such that the present keyphrases were in the order they appeared in the article, and absent keyphrases followed the present keyphrases in a random order.[3]

We trained the model on one GPU on a g4dn instance on AWS for 100,000 steps, a process that took 48 hours. We decoded the keyphrases using a beam search of width 50, and took the union of all keyphrases generated in the top 50 sequences, results shown in Table 3. We also performed a greedy decoding (results shown in the midterm report).

As seen in Table 3, we reproduce the results found in the original paper, with some metrics being slightly lower than the reported since the original researchers were able to train for 300,000 steps, as opposed to 100,000 steps.

5.1.4 Baseline Training Procedure on KPTimes

We also trained CatSeqD-Transformer on the KPTimes dataset. The same vocabulary file used to train KP20k was used to train KPTimes, a mistake we made. Still, we were able to produce results comparable to reported benchmarks (see Results section). Word vector embeddings were initialized in the same way. The ground truth keyphrases were ordered in the same way. We trained the model on one GPU on a g4dn instance on AWS for 70,000 steps. Training took 60 hours, longer than the prior experiment on KP20k because the news articles were on average around 750 words longer than the abstracts in KP20k. Evaluation metrics are reported in the Results section.

5.2 Model Extension: BART

5.2.1 Model Extension Rationale

Present ($\mathbf{F_1@10}$)		Absent ($\mathbf{R@50}$)	
Ours	**BERT-KPE**	**Ours**	**BERT-AKG**
28.9	43.7	11.0	28.2

Table 4: Comparing the performance of our CatSeqD-Transformer (Ours) with the BERT-PKE (present keyphrase extraction) model and BERT-AKG (absent keyphrase generation) model from [11]

The logical extension to the baseline transformer model is to use pre-trained token embeddings from a state-of-the-art language model, rather than randomly initialized token embeddings. A recent study used pretrained BERT embeddings to train two models, one explicitly for present keyphrase

[2]For a more extensive discussion on these regularization techniques, see our midterm report 2.2.2 The Challenge of Generating a Variable Number of Keyphrases.

[3]A review paper [13] showed that the order of the ground truth keyphrases does matter, and that the present-absent ordering generally leads to the best results.

extraction and another for absent keyphrase generation, both of which outperform our baseline on KP20k (see Table 4). We settled on BART [10] since it naturally combines the bidirectional encoder of BERT, which we thought would be reasonable for encoding news articles, with the autoregressive decoder of GPT, which can be used for decoding both present and absent keyphrases. Furthermore, there existed a version of BART finetuned on CNN news article summarization. We call our model extension **CatSeqD-BART-CNN**.

Choosing BART led us into a rabbit hole of incorporating many of the enhancements that BART provided, the first of which is that BART (the large version) is twice as large as our CatSeqD-Transformer model. It has double the number of encoder and decoder layers, double the number of attention heads, and double the size of token embeddings. Second, BART uses the RoBERTa subword tokenizer rather than spaCy's word-level tokenizer. Since subword tokens are more capable of composing rare words and named entities, we did not use a copy mechanism as in the baseline model.[4] See Table 3 above for a comparison between the baseline and model extension.

As for the pretrained token embeddings, we downloaded bart-large-cnn from Hugging Face [10]. This is a version of BART that was fine-tuned on 300,000 CNN Daily Mail news articles for the summarization task. Keyphrase generation is a very similar task to summarization, and furthermore, this BART model was trained in the same domain, so is well-suited to our problem.

5.2.2 Model Extension Implementation

BART has already been implemented for us in Fairseq, an open-source sequence modeling toolkit from Facebook [16]. We reached out to Rui Meng, the primary author of the original neural keyphrase generation model CopyRNN [14], for advice about adapting BART to our task. It turned out that he had already added a custom task in Fairseq for keyphrasification, which parses the raw input documents, and primes the BART model to generate sequences of keyphrases. We made a fork of his repository, and added a training script which would incorporate bart-large-cnn our pretrained model, and KPTimes our dataset. The private repository, which has been shared with the instructors of 11-711, can be found here: https://github.com/eric-zhizu/fairseq-kpg.

5.2.3 Training Procedure on KPTimes

The CatSeqD-BART-CNN model was fine tuned on the KPTimes dataset. Rui Meng provided us with a subword dictionary file, which was created using byte-pair encoding (BPE) according to the RoBERTa subword tokenizer on the MagKP scientific research dataset (a larger version of KP20k) [13]. Again, in hindsight, we should have generated these vocabulary files ourselves on our particular news dataset, but this did not seem to hinder the model from outperforming published results.

During data pre-processing, ground truth keyphrases were ordered in the same way as in the baseline model — present keyphrases first followed by absent keyphrases. However, we made a mistake when separating the keyphrases from one another. In the KPTimes dataset, roughly 8% of keyphrases are a sequence of synonym separated by semicolons, such as "NYC; New York City" or "Marijuana; Pot; Weed." An error in our data pre-processing caused these synonyms to be concatenated and considered as one keyphrase, such as "NYC New York City" and "Marijuana Pot Weed." Nonetheless, from human qualitative evaluation, the model was able to learn the mistakes and generate these sequences of synonyms successfully.

We trained the model on a single GPU with 16 GB of CUDA RAM on a g4dn.2xlarge instance on AWS for 80,000 steps, a process that took 100 hours. We decoded the keyphrases using a beam search of width 10, and took the union of all keyphrases generated in the top 10 sequences.

6 Results

6.1 Quantitative Results

The first experiment we ran was to use the baseline CatSeqD-Transformer trained on the KP20k dataset, to generate keyphrases for articles in the KPTimes test dataset to see if the model is able

[4] See 2.2.1 Generating Single Keyphrases in the midterm report for a thorough discussion of the baseline model's copy mechanism.

to transfer well from the science domain to the news domain. The results were poor for reasons discussed in the Datasets section. The model was practically unable to generate absent keyphrases, and it severely underperformed the benchmark results published by the paper which released the KPTimes dataset, as seen in Table 5.

Our second experiment was to train the baseline model CatSeqD-Transformer on the KPTimes dataset explicitly. This resulted in a F_1 metric comparable to the benchmark, and vast improvements over the model trained on KP20k.

Model	Present ($\mathbf{F_1@\mathcal{O}}$)	Present ($\mathbf{F_1@10}$)	Absent ($\mathbf{R@50}$)	Total ($\mathbf{F_1@10}$)
Benchmark CopyRNN from [5]	N/A	N/A	N/A	39.3
CatSeqD-Transformer Trained on KP20k	13.37	15.10	0.80	11.75
CatSeqD-Transformer (Baseline)	53.12	41.51	51.13	38.04
CatSeqD-BART-CNN (Extension)	56.07	49.09	63.75	41.68

Table 5: The results of evaluating our models (CatSeqD-Transformer and CatSeqD-BART-CNN) on KPTimes compared to benchmark results published by the paper which released the KPTimes dataset (CopyRNN) [5]

Our third experiment was to train the model extension CatSeqD-BART-CNN on the KPTimes dataset. This resulted in a 3% increase of the $F_1@\mathcal{O}$ metric, a 7% increase of the $F_1@10$ metric for present keyphrases, as well as a 12% increase of the $R@50$ metric for absent keyphrases. Compared to the benchmark, there was a 2% increase of the $F_1@10$ metric for both present and absent keyphrases.

The fact that the model extension resulted in a larger improvement for the $F_1@10$ metric than the $F_1@\mathcal{O}$ metric indicates that the model is not only producing more correct keyphrases, but it is also ranking the correct keyphrases higher. In the top 10 keyphrases of its output (measured by $F_1@10$), it makes less incorrect keyphrases, and outputs more correct keyphrases, albeit $F_1@\mathcal{O}$ is probably a better metric since the articles only have on average five keyphrases each [5].

The improvement over the recall of absent keyphrases as well, we hypothesize, is due to the subword tokenization scheme of BART. The original transformer model CatSeqD-Transformer relies on a copy mechanism to generate keywords not in its vocabulary but found in the source document, whereas the extended model can use subword tokens to generate rare words not found in the source document.

Notably, CatSeqD-Transformer had peak performance at around 25,000 steps in predicting present keyphrases.[5] However, CatSeqD-BART-CNN, which trained for 80,000 steps did not peak, and seemed to continue to increase both its metrics on present and absent keyphrases. If we had the time, we surely would have continued training the BART model.

6.2 Qualitative Analysis

After manually reading through a subset of the results, here are a couple observations on what CatSeqD-BART-CNN and CatSeqD-Transformer do well:

Generate correct absent keyphrases not present in the ground truth: For example, in an article containing the ground truth keyphrase "train wreck" absent from the source article, the CatSeqD-Transformer model generates "public transit" and "accidents and safety" despite those not being in the ground truth or the source document. The CatSeqD-BART-CNN model does generate "train wreck", but also generates "fatalities and casualties" as well as "public transit."

Predict important named entities: Many articles have the names of celebrities (like Herman Edwards) or organizations (like the Washington Metropolitan Area Transit Authority) in their ground truth keyphrases. Both models are able to generate them.

BART corrects for permutations of named entities: Despite the above, CatSeqD-Transformer sometimes attempts to overcompensate. In an article about murder with the names of several victims, the CatSeqD-Transformer model generates permutations of the people's names without even

[5] On absent keyphrases, the metrics for both models continually increased until the last step trained.

generating the keyphrase "murder." The CatSeqD-BART-CNN model, on the other hand, correctly ignores the names of victims, and predicts "murder" correctly.

Infers absent keyphrases: In an article that mentions "sexual abuse of children", both models correctly output the ground truth labels "child abuse" and "rape", which are absent in the article.

Correctly reads narrative voice: Some articles are written entirely in a narrative voice, like a work of short fiction, rather than in a neutral expository voice. In those cases, both models are still able to correctly extract the ground truth keyphrases (and relevant non-ground truth keyphrases).[6]

7 Error Analysis

We analyze the errors by two parts: one is from the comparison of the baseline model on KPTimes and KP20k; the other is from comparison between the baseline and our proposed extension on KPTimes.

7.1 Domain-specific Errors

One unique error that CatSeqD-Transformer makes on KPTimes is that the model keeps inserting or appending words to shorter keyphrases to create new ones with more tokens. This error fails at two aspects—repetition and boundary of meaning. The overly long keyphrases contains either repeated or similar meaning tokens. For example, one news article outputs "new york city management council", "new york city management inspection management", "new york city management management council", "new york city management inspection management district", and "new york city management inspection management district mamhatten" in the keyphrases list. We think that this type of error is from the model trying to make up the named entities with so much details provided in the news article. CatSeqD-Transformer makes such errors on the KP20k less severely because first the abstract are already trimmed out many unimportant details and second there aren't as many variations in the names as in the news articles. We also notice this error less frequently in the output of CatSeqD-BART-CNN, probably because it lacks a copy mechanism, so it is not trying to copy large swaths of named entities from the source document.

7.2 Model-specific Errors

CatSeqD-BART-CNN learns to create named entities much better than the baseline. From a quick look through the results, phrases like "karen marie", "karen marie jacks", "karen marie lydie", "karen marie fenty", "karen marie y marie", which are generated by the copy mechanism in the CatSeqD-Transformer are not found in the generated keyphrases of CatSeqD-BART-CNN. However, the BART model truncates the keyphrases at wrong tokens, such as after "of" or "and". This indicates that the model actively attempts to summarize groups of meanings appearing in the article (represented by ground truth keyphrases like "appointments and executive changes", "morbidity and mortality", and "computer and the internet"), but fails to learn the functions of proposition or conjunction words.

Other errors are made in the process of trying to figure out abbreviations of either organizations, places, or people. For example, the model sometime extract single letter as keyphrases. We guess that these are from people's middle names. The model also creates other ambiguous truncated keyphrases, like "Medic", "Imm", or "Il". We suspect that this type of error is a side-effect of using the byte-pair encoding (BPE) for subword tokenization instead of work-based tokenization. BPE helps reducing the repetition and overlength errors, but occasionally outputs subword tokens common in multiple words in the article as keyphrases. Nonetheless, these erroneous keyphrases are ususally ranked low in the list.

[6]For instance, one article begins, "Make his tie green instead of red and ut the room at Hofstra University instead of Arrowhead Stadium This could have been a Herman Edwards news conference with the Jets Edwards stood behind a lectern.." The closest that the article comes to a topic sentence is, "he seemed to have an answer for everything except what everyone wanted to know Why was he here being introduced as the Kansas City Chiefs head coach just six days after." And yet, both models correctly predict both the important named entities like "Jets" and "Kansas City Chiefs" as well as topical keyphrases that don't appear in the ground truth such as "sports drafts and recruits."

8 Conclusion and Future Directions

To sum up, we extended a state-of-the-art keyphrase generation network to use pretrained language model subword token embeddings, and raised the benchmark information retrieval metrics by 2-12%. We show some of the advantages and disadvantages of the keyphrase generation models in our qualitative analyses and demonstrate that the extended model, which uses subword tokens instead of a copy mechanism, is able to avoid some of the pitfalls with named entity keyphrases to which the baseline model falls is susceptible.

We are encouraged by the results CatSeqD-BART-CNN and CatSeqD-Transformer have shown on articles that have a more narrative as opposed to expository style. One immediate short-term follow-up to this study that we plan to do is to train these architectures on a set of 5,000 short stories curated by WritingAtlas.com.

Many extensions could be made to our experiments as well. On the data pre-processing side, we would like the model to take into account both punctuation and line breaks. Many of the news articles without these elements become even hard for humans to read and understand. Secondly, we would like to try a variant of BERT such as Longformer in order to better process long news articles. The average word length of an article is 921 words [5], and the model currently truncates the input at 512 tokens.

We would also like to address some of the errors found in our qualitative analysis by incorporating a named entity recognition module, or an incomplete phrase detector, in order to filter out obviously ill-formed keyphrases. Another promising extension might be incorporating kNN in neural language model [7] during the decoding of keyphrases. kNN could be particularly helpful in the news domain because multiple news articles may be written over the course of an event. Providing the model with information from even a small number of k nearest neighbors could ameliorate the lack of assumed knowledge that naturally come with human readers.

9 Acknowledgements

We would like to acknowledge Rui Meng for his help and guidance throughout the project. Both our baseline and our model extension are built off of GitHub repositories that he authored or co-authored. He additionally dedicated time to help us debug the code and training.

We would also like to thank the course instructors of 11-711 for their help during office hours, and for giving us valuable advice.

References

[1] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null):993–1022, March 2003.

[2] Jonathan Chang, Sean Gerrish, Chong Wang, Jordan Boyd-graber, and David Blei. Reading tea leaves: How humans interpret topic models. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.

[3] Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. Keyphrase generation with correlation constraints. *arXiv preprint arXiv:1808.07185*, 2018.

[4] Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R Lyu. Title-guided encoding for keyphrase generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6268–6275, 2019.

[5] Ygor Gallina, Florian Boudin, and Beatrice Daille. Kptimes: A large-scale dataset for keyphrase generation on news documents. *arXiv preprint arXiv:1911.12559*, 2019.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[7] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019.

[8] Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. SemEval-2010 task 5 : Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[9] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada, July 2017. Association for Computational Linguistics.

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

[11] Rui Liu, Zheng Lin, and Weiping Wang. Keyphrase prediction with pre-trained language model. 04 2020.

[12] Rui Liu, Zheng Lin, and Weiping Wang. Keyphrase prediction with pre-trained language model. *arXiv preprint arXiv:2004.10462*, 2020.

[13] Rui Meng, Xingdi Yuan, Tong Wang, Sanqiang Zhao, Adam Trischler, and Daqing He. An empirical study on neural keyphrase generation. *arXiv preprint arXiv:2009.10229*, 2020.

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

[15] Yishu Miao, Lei Yu, and Phil Blunsom. Neural variational inference for text processing. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1727–1736. JMLR.org, 2016.

[16] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[17] Akash Srivastava and Charles Sutton. Autoencoding variational inference for topic models. In *ICLR*, 2017.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[19] Xiaojun Wan and Jianguo Xiao. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI'08, page 855–860. AAAI Press, 2008.

[20] Rui Wang, Deyu Zhou, and Yulan He. Atm: Adversarial-neural topic model. *Information Processing Management*, 56:102098, 11 2019.

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

[22] Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Peter Brusilovsky, Daqing He, and Adam Trischler. One size does not fit all: Generating and evaluating variable number of keyphrases. *arXiv preprint arXiv:1810.05241*, 2018.