Computational Linguistics and Intellectual Technologies:
Proceedings of the International Conference “Dialogue 2019”

Moscow, May 29—June 1, 2019

NEWS HEADLINE GENERATION USING STEMS, LEMMAS AND GRAMMEMES

Stepanov M. A. (projectttower@gmail.com)
MIPT, Dolgoprudny, Russia

Headline generation is a task that has a good solution based on seq2seq models with an attention mechanism. However, it is still quite challenging to deal with morphologically rich languages, such as Russian, which have many word forms and therefore larger vocabularies. To deal with complex dependencies arising in such languages we propose several approaches based on using stems and grammemes. We applied these approaches to the pointer-generator network and took second place in the competition on headline generation held by the conference Dialogue-2019.

Key words: headline generation, Russian language, pointer-generator, stem, flexion, lemma, grammeme

ГЕНЕРАЦИЯ ЗАГОЛОВКОВ НОВОСТНЫХ СТАТЕЙ, ИСПОЛЬЗУЮЩАЯ СТЕМЫ, ЛЕММЫ И ГРАММЕМЫ

Степанов М. А. (projectttower@gmail.com)
МФТИ, Долгопрудный, Россия

Задача генерации заголовков имеет хорошее решение, которое базируется на использовании seq2seq моделей с механизмом внимания. Однако в случае морфологически богатых языков таким моделям приходится сталкиваться с более сложными зависимостями, которые могут проявляться в виде большого количества словоформ и их сочетаний друг с другом. Мы предлагаем несколько подходов, которые могут помочь автоматическим seq2seq генераторам заголовков учитывать зависимости таких языков, как русский. Мы также применили данные подходы к архитектуре генератора-указателя и заняли второе место на соревновании по генерации заголовков, проведённом в рамках конференции Диалог-2019.

Ключевые слова: генерация заголовков, генератор-указатель, стемминг, флексия, лемма, граммема

1. Introduction

There are two main groups of text summarization approaches: abstractive and extractive. While extractive aproaches try to find the most informative subset of the text and copy it, abstraction-based systems generate words and phrases not from the source, but from the vocabulary, using learned natural language dependencies.

Automatic headline generation is a type of the summarization task. The aim of summarization is to create a shorter version of the text (in our case, the title), which contains the main idea of the given article. Working on task of generating headings has an advantage over the traditional summarization: it is much easier to find articles with titles than with annotations, which is very convenient for systems based on machine learning methods. There is almost an infinite supply of news articles in all major languages and almost all of them have a headline.

But, despite the existence of a huge amount of data, headline generation system still should be able to deal with dependencies of natural language, and the creation of this system is a challenging task. Due to this difficulty the vast majority of past decisions use extractive methods (see [1] or [2]), but the relatively recent success of sequence-to-sequence models [3] has made the abstractive approach viable (see [4] or [5]). Now it is possible to automatically read and generate text that has the structure similar to the headings written by human.

However, the benefits that seq2seq brought were not enough to create desirable headlines: these systems have problems such as the words repeating and the inability to use out-of-vocabulary (OOV) words of a source article. To enable OOV extraction, a pointer-generator model has been developed and introduced by See et al. [6]. This model is both extractive and abstractive: it is based on seq2seq, but can copy words from text too. Additionally, the coverage mechanism and the usage of a coverage loss (penalty for repeating words) during the training phase makes this model less prone to repetition. Due to these advantages, we chose the pointer-generator network with coverage mechanism as the baseline.

Though pointer-generator network can create human-like headlines of English news, it is quite difficult for the model to achieve the same success with, for example, Russian articles. Even simple vocabulary of morphologically rich language can contain several million forms and variations. For a model it is harder to find suitable words in the space of possible variants expanded by word forms. With a larger vocabulary it takes much more memory, computing power and time to teach the network to generate desirable headings.

In this paper we propose several approaches to deal with problems of morphologically rich languages: stem+flexion encoding and grammeme embeddings. We also present results of experiments that were made with RIA corpus[1] (presented by [7]) and Lenta corpus[2] during the competition track on the headlines generation held by the conference Dialogue-2019.

[1] https://github.com/RossiyaSegodnya/ria_news_dataset

[2] https://github.com/yutkin/Lenta.Ru-News-Dataset

2. System description

2.1. Baseline model

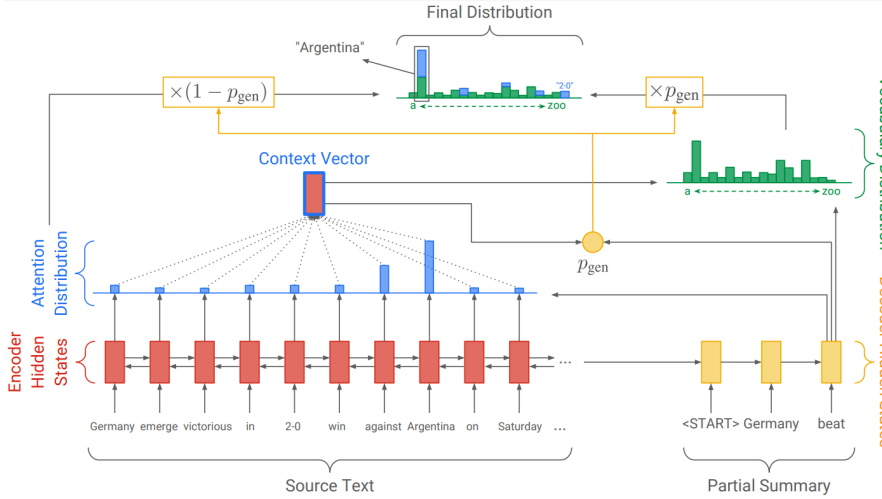

Figure 1: Pointer-generator model scheme from [6]

The pointer-generator network is based on a sequence-to-sequence model with an attention mechanism (**Figure 1**). It uses the encoder to make encoder hidden states h_i, which store the extracted information from the article. Article tokens w_i are fed one-by-one to the encoder's embedding layer and the single-layer bidirectional LSTM. After that the model generates words of abstract step by step, applying the decoder (unidirectional one-layer LSTM) to produce a decoder state s_t from an embedding of previously generated word y_{t-1} and so-called context vector (created by the attention mechanism) h_t^*. Then the network gets the output vocabulary distribution (that show which word is most probable as next token of the headline) from the decoder state.

The attention mechanism is a modification of the seq2seq model which helps the decoder to produce the next word indicating which words of the source article are the most important at the step t. This information is contained in the attention distribution a^t calculated by this mechanism. Next, using the attention distribution as weights in the sum of encoder states h_i, model creates context vector h_t^*—the "second ingredient" of the output vocabulary distribution.

In addition to the generation of words from the fixed vocabulary this model is able to copy tokens from the source article. It is realized by calculating the generation probability p_{gen} at each step t. Then the network use p_{gen} as a soft switch to choose between generating a word using the vocabulary distribution, or copying a word from the text using a^t, which shows the most suitable tokens for extraction. This modification makes model both extractive and abstractive and therefore more flexible for different kinds of situations.

To cope with the output repetition problem, coverage mechanism is also involved in the title generation process. This modification retains all attention distributions produced by the model at each step t, and gives an additional loss if the model use similar a^t. If pointer-generator is trained with coverage mechanism, it is more liable to extract different words from the source and use different context vectors, which makes the model less repetitive[3].

2.2. Stem+flexion encoding

In order to help model to work with larger vocabulary of morphologically rich languages, we experimented with two approaches. Both of them change the structure of input and output words to make the vocabulary sufficiently smaller with no drops in performance.

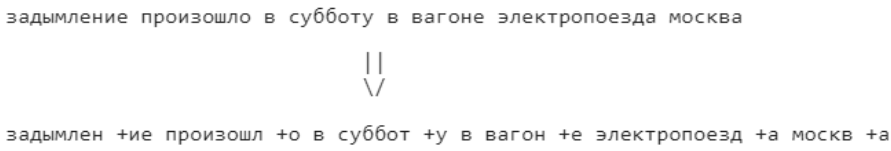

Figure 2: Example of stem+flexion encoding

The first approach is based on encoding each word as a pair of its stem and flexion (or only stem if there is no flexion). To encode n words with m forms of each word model can work with a list of n stems and a fixed number of flexions instead of a vocabulary with $n * m$ words, which makes it easier for a network to find natural language dependencies in articles. The output of model consists of stems and flexions too and it can be easily decoded into words sequence.

In our experiments we use a Porter stemmer[4] [8] for automatic encoding and a vocabulary of stems and flexions with 450 flexions[5]. Each flexion has a '+' as a prefix to distinguish them from stems (**Figure 2**) and to restore headline from the output sequence of stems and flexions.

It is important to mention that we don't make any changes in the model architecture in this part of experiments, only changes in input and output processing. But we also make attempts to use 3-layer encoder and decoder instead of single-layer to help the model to learn more sophisticated dependencies[6].

---

[3] read [6] to get more information about baseline model

[4] https://medium.com/@eigenein/стеммер-портера-для-русского-языка-d41c38b2d340

[5] https://colab.research.google.com/drive/1DEEwaFGQV6-SvoBuqalvxrSL3uLqI535

[6] https://colab.research.google.com/drive/1U6BHW2TgfnjpxnoSdJzWLf0_23mVZFqF

2.3. Grammeme embeddings

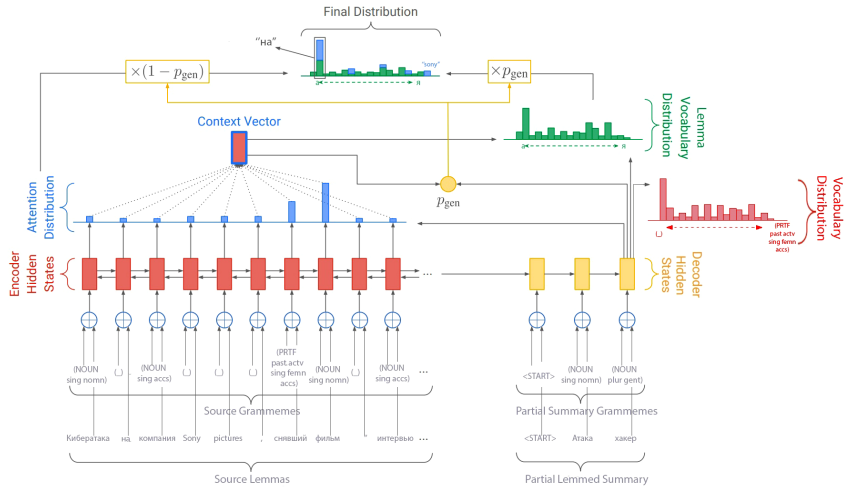

Figure 3: Pointer-generator model using grammemes

Another approach is based on the usage of lemmas and grammemes instead of words. We use a morphoparser (we choose pymorphy2[7]) to divide each word into its lemma and a string consisting of a part of speech and all values of changeable grammatical values (**Figure 3**): For example, a noun 'хакеров' is encoded to lemma 'хакер' and string '(NOUN plur gent)'. If a part of speech is not changeable (prepositions, conjunctions), then word gets string '(_)'. With this method, we created the vocabulary of lemmas and the vocabulary of strings with grammemes, which in our experiments has a size of 300.

We have changed the model architecture for these experiments: instead of the embedding layer for tokens of encoder's and decoder's input sequences we have made two independent layers for lemmas and grammeme strings. Network transforms article words to two sets of lemmas and grammemes and each of them passes through its own embedding layer. Then model concatenates two embeddings and gives the result to encoder and decoder. In addition to the vocabulary distribution (of lemmas) decoder generates distribution over the vocabulary of strings with grammemes mentioned above.

Next, in the training phase model calculates the loss. We have included additional cross entropy loss for the grammeme output sequence in order to help the network to learn how to create right word forms. If the title generator works in the test phase, it tries to create headline with morphoparser by applying grammeme strings to lemmas (if it is impossible, the model gives lemma to output)[8].

---

7 https://pymorphy2.readthedocs.io/en/latest/

8 https://colab.research.google.com/drive/1zIJ3Pk1oljRR8qTaZn25UkfDTeTKIL77

3. Data and training

We consider two corpora: RIA and Lenta datasets. RIA dataset was provided by Russian news agency "Rossiya Segodnya" and used in the competition track of the conference Dialogue-2019[9]. It contains 1,003,869 news articles of the time period from January 2010 to December 2014. We use this corpus as a training dataset which has an additional preprocessing such as cleaning from html-tags, lower-casing and tokenization. To speed up the learning of models, articles are also processed by the stem+flexion encoder and divided into lemmas and grammemes sequences by the morphoparser.

Lenta corpus has 739 new articles from 1999-08-30 to 2018-12-15. We chose 10,000 random articles to form the test dataset. These texts were preprocessed in the same way as the train dataset.

4. Experiments

4.1. Models

In this work there were 4 different models which trained on Ria Corpus and were tested on Lenta Corpus. Here they are: **baseline pointer-generator**, **pointer-generator using stems**, **pointer-generator using stems and 3-layer LSTM** and **pointer-generator using grammeme embeddings**.

4.2. Training

The models trained with the Adam optimizer using a scaled learning rate. All of them worked with vocabularies with 100,000 tokens and used token embeddings with the size of 128. Grammeme embeddings had the size of 32. Embedding layers were shared between encoder and decoder for all models. The size of the hidden vectors of LSTM layers was equal to 256. In addition, the length of the input sequences was limited with 600 tokens for the model with stems and 400 for other models. Reference headlines were also truncated to 20 (for the model with stems) and 12 tokens (for other models). For headline generation, beam-search size was made equal to 4.

All models trained with batches of articles with the size of 32. Baseline pointer-generator trained for 400,000 epochs, as models working with stems. Model with grammemes embeddings passed through 285,000 training epochs.

5. Results

We present our results on Lenta dataset in the Table 1. As it can be seen, all models with modifications surpassed vanilla pointer-generator on ROUGE-1, ROUGE-2 and ROUGE-L F1 scores. Model with 3-layer LSTM shows better results than the same

[9] https://vk.com/@headline_gen-announcement

model but with single-layer encoder and decoder. Both of them had the same number of training epochs, so it seems, that more complex architecture helps title generator to understand natural language dependencies arising in this dataset better.

Network using grammeme embeddings has better R-1 and R-L scores than models with stems, but it loses in R-2 scores. But this model has single-layer LSTM, and if encoder and decoder would be multi-layer, the network with grammeme embeddings could outperform models with stems in all scores and with a large margin, what makes usage of grammemes more preferable than applying stem+flexion encoding.

Table 1: ROUGE-1,2,L F1 and recall scores, on Lenta corpus

Model	R-1-f	R-1-r	R-2-f	R-2-r	R-L-f	R-L-r
Pointer-generator (baseline)	21.36	22.27	8.69	8.70	19.25	20.79
Pointer-generator with stems	23.47	23.81	10.24	10.39	21.24	22.27
Pointer-generator with stems and 3-Layer LSTM	25.16	25.82	11.32	11.63	22.78	24.13
Pointer-generator with grammeme embeddings	25.23	25.79	10.33	10.60	22.82	24.08

Using the model with stems, we took second place in headline generation contest held by Dialogue-2019. This model was evaluated on the private part of the RIA dataset and had a score of 20.29 (mean of R-1-f, R-2-f, R-3-f). Unfortunately, 3-layer stem model and the model with grammeme embedding didn't participate in competition because of lack of training time at the end of this event.

Additionally, we present headlines generated by all four models with two random texts from the dataset (Table 2).

Table 2: Samples of headlines generated by models

Original text, truncated: дамаск , 11 мая . - риа новости . президент россии дмитрий медведев считает опасным дальнейший рост напряженности на ближнем востоке . " дальнейший разогрев ситуации на ближнем востоке чреват взрывом и катастрофой " , - сказал медведев на пресс-конференции по итогам переговоров с президентом сирии башаром асадом . " с моей стороны было специально подчеркнуто , что россия будет и дальше предпринимать все от нас зависящее для того , чтобы помогать восстановлению арабо-израильского мирного процесса на основе международно-правовой базы , которая имеется ...
Original headline: медведев : " разогрев " ситуации на ближнем востоке чреват катастрофой
Headline by baseline pointer-generator: медведев считает опасным дальнейший рост напряженности на ближнем востоке
Headline by pointer-generator using stems: медведев : рост напряженности на ближнем востоке чреват взрывом
Headline by pointer-generator using stems and 3-layer LSTM: напряженность на ближнем востоке опасна , заявил медведев

Headline by pointer-generator using grammeme embeddings: медведев считает опасным рост напряжённости на ближнем востоке
Original text, truncated: москва , 5 мая - риа новости . задымление произошло в субботу в вагоне электропоезда москва - фрязино ярославского направления московской железной дороги , из-за чего пассажиров пришлось пересадить в другую электричку , сейчас движение поездов восстановлено , сообщил риа новости руководитель службы корпоративных коммуникаций мжд владимир мягков . “ сегодня в районе платформы чкаловская в электропоезде номер 6707 в пятом вагоне произошел нештатный разогрев буксы колесной пары , что дало небольшое задымление . в связи с этим электропоезд был остановлен ” , - сказал мягков ...
Original headline: в электричке в подмосковье произошло задымление вагона
Headline by baseline pointer-generator: задымление произошло в электричке на подмосковном железной дороге
Headline by pointer-generator using stems: задымление произошло в вагоне поезда москва - фрязино ярославского направления
Headline by pointer-generator using stems and 3-layer LSTM: задымление произошло в вагоне электропоезда московской железной дороги
Headline by pointer-generator using grammeme embeddings: задымление произошло в вагоне электрички мжд - фрязино

6. Conclusion

In this paper, we explore the application of two approaches to the pointer-generator network processing, such as usage of stems and grammemes, and with these described modifications model outperforms its own results on Russian news articles. The future work will focus on testing models on other datasets and experimenting with settings and subsystems of the model.

Acknowledgements

Author is thankful to Ivan Smurov for useful discussions and proofreading, organizers of competition track of Dialogue-2019 on headline generation for providing test dataset and interesting task.

References

1. *Julian Kupiec, Jan Pedersen, and Francine Chen* (1995). A trainable document summarizer. In International ACM SIGIR conference on Research and development in information retrieval
2. *Horacio Saggion and Thierry Poibeau* (2013). Automatic text summarization: Past, present and future. In Multi-source, Multilingual Information Extraction and Summarization, Springer, pages 3–21.

3. *Ilya Sutskever, Oriol Vinyals, and Quoc V Le* (2014). Sequence to sequence learning with neural networks. In Neural Information Processing Systems.
4. *Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang* (2016). Abstractive text summarization using sequence-to-sequence RNNs and beyond. In Computational Natural Language Learning.
5. *Alexander M Rush, Sumit Chopra, and Jason Weston* (2015). A neural attention model for abstractive sentence summarization. In Empirical Methods in Natural Language Processing.
6. *Abigail See, Peter J. Liu, Christopher D. Manning* (2017). "Get To The Point: Summarization with Pointer-Generator Networks" arXiv:1704.04368.
7. *Daniil Gavrilov, Pavel Kalaidin, Valentin Malykh* (2019). "Self-Attentive Model for Headline Generation" arXiv:1901.07786.
8. *Martin F. Porter* (1980). An algorithm for suffix stripping.