Auto Generation of Arabic News Headlines

Yehia Khoja Omar Alhadlaq Saud Alsaif

December 16, 2017

Abstract

We describe two RNN models to generate Arabic news headlines from given news articles. The first model is a standard sequence-to-sequence architecture, and the second model is a sequence-to-sequence model with attention. We train the models on a dataset of 30K pairs of articles and headlines. We show that the two models perform similarly, as they both can generate relevant headlines, but sometimes with made up details.

1 Project Overview

We want to create a model that takes an Arabic news article as input, and then generates a headline for that article as an output. As such, our project will be a deep learning effort that aims to use a sequence-to-sequence (seq2seq) model, which showed success in machine translation (Luong et al., 2017, 2015), since our problem can be casted as a machine translation problem. We are using a publicly available data set containing 31,030 Arabic news articles with their respective headlines. The data is structured into multiple `.json` files, where each file contains around 2,000 articles.

The outline of this report is as follows. In Section 3, we present the data and our preprocessing approach. In Section 4, we present the sequence to sequence model that we use and the additional attention mechanism. In Section 5, we present the results of training the models, along with some examples and discussions from the outputs. Finally, in Section 6 we present our conclusions and future work.

2 Related Work

Recently there has been many advances in neural machine translation using RNN architectures like seq2seq and attention (Luong et al., 2017, 2015; Gu et al., 2016; Sutskever et al., 2014). Casting the problem of generating headlines as a machine translation problem was presented in the work done by Lopyrev (2015), where he generated English news headlines using a seq2seq model with attention. Lopyrev (2015) had good results in terms of the quality of the generated headlines. We want to achieve similar performance with Arabic headlines. One thing to note is that our data set has 31K articles while Lopyrev (2015) had a data set of 5.5M articles.

source	The newspaper
url	URL to the article
date_extracted	A timestamp of the date of the article
title	Headline of the article (can be empty)
author	Author of the article (can be empty)
content	Body of the article

Table 1: The fields of each data point, we are concerned with the source, content, and the title.

3 Data Management Approach

3.1 Data Source

We use a dataset of 31,030 Arabic news article and headline pairs in the Saudi Newspapers Arabic Corpus (Alhagri, 2015). The articles are written in Modern Standard Arabic (MSA), and were extracted from 14 different newspapers covering topics like politics, local news, culture, and sports. Table 1 shows the information given for each article in the dataset.

3.2 Data Clean Up

The Arabic language does not have any vowels, and it uses annotation markers on the letters to identify the correct pronunciation of each letter. Hence, each word can be written in multiple forms that computers can not tell if they are the same word or not. Therefore, our first task at cleaning up the data was to normalize all words by removing all annotations, which are considered separate characters, from the data. We demonstrate an example for reference in Figure 1.

طَالِبٌ طالب

(a) Total of 7 characters (b) Total of 4 characters

Figure 1: The Arabic word for "Student" with annotation (left) and without annotation (right)

Second, we also separate all punctuation marks, such as commas and fullstops, from words and considered each punctuation mark as a separate token.

3.3 Data Preprocessing

Beyond cleaning up the data, we have also preprocessed the data to reduce the size of the computations required. Namely, instead of representing each article as a matrix whose columns are one hot vectors that represent each word, we represent the article as a vector of indices, where each index is the index of the word in the vocabulary. Later we use these indices to refer to the embedding of the word in the word embeddings matrix.

In addition, based on our initial observations with processing the data, we noticed that the length of our vocabulary vector has exceeded 270k words. Hence, in order to speed up the initial training, we have decided to limit our vocabulary to the 15k most frequent words between all articles, which we later increased to 100k.

Moreover, we have created three special tokens to incorporate them in each article:

- PAD TOKEN: used to pad articles shorter than the chosen max article length, which we will explain in the next section
- UNK TOKEN: used for words that aren't in the vocabulary
- SOS TOKEN: used to signal the start of the article
- EOS TOKEN: used to signal the end of the article

3.4 Data Split

We have split the original raw data into train, validation, and test sets. For each set, we have two text files: one file containing the news articles in the set with one article per line, and the second file containing the headlines of each article with one headline per line. We have divided the 31,030 articles we had as follows: 80% in the train set, 8% in the validation set, and 12% in the test set. In order to keep the distribution of the sources consistent among the train, dev, and test set, we performed the split on each of the 14 newspapers separately.

4 Modeling Approach

4.1 Standard Model: Sequence to Sequence

We use a sequence to sequence architecture (seq2seq) similar to the one introduced in Sutskever et al. (2014); Luong et al. (2015). The seq2seq model comprises two components, an encoder and a decoder. Each one has an embeddings layer that converts the input/output texts into vectors. First, we have our encoder. The encoder is a multi-layer RNN architecture that maps a sequence of word embeddings input to an encoded representation of the whole sequence, which then is used by the decoder to produce the output. We use LSTM as our basic RNN cell, in both the encoder and the decoder, given its effectiveness in dealing with problems with long sequences (Hochreiter & Schmidhuber, 1997). The decoder takes the output of the last hidden state of the encoder as input in addition to the actual headline (only in training, in inference we use last word generated in the title as input). In addition to the embeddings layer and the two hidden layers, the decoder has a projection layer that outputs a softmax vector over the vocabulary, used to generate each word in the headline. Figure 2 shows the structure of this model.

4.2 Improved Model: Seq2Seq with Attention

As we saw in the standard seq2seq model, the encoder compresses the input into a single-vector encoded representation of the input, potentially leading to the loss

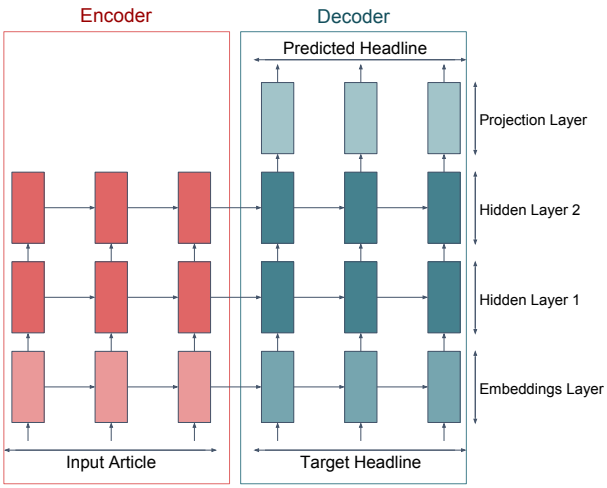


Figure 2: A seq2seq model with an encoder and decoder

of important information that might be needed in the decoder. Using an attention mechanism is one known solution to this problem. In our attention model, we attend to different parts of the input in every decoder step, allowing the model to empasize these "essential" parts of the input in its encoded representation. The attention we use in our model is the one described in Luong et al. (2015), which is also used in a several state-of-the-art systems. As shown in Figure 3, at each decoder time step, decoder hidden state output is compared with all hidden state output of the encoder to compute the attention weights. The encoded representation of the input is then the weighted average of the encoder hidden state outputs, instead of only being the output of the last hidden state as in the standard seq2seq model.

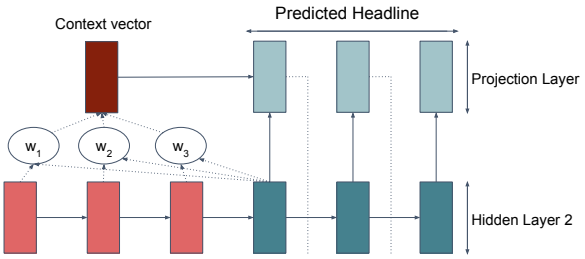


Figure 3: Structure of a seq2seq model with attention, we only show the top parts of the seq2seq model for brevity

4.3 Optimization and evaluation metrics

To optimize the weights of the model, we used batched stochastic gradient descent. We selected our loss metric to be the cross entropy between each word in the predicted headline and the corresponding word in the actual headline.

$$CE(y, \hat{y}) = -\sum_{i=1}^{|V|} y_i \log \hat{y}_i \tag{1}$$

where $|V|$ is the size of the vocab, y is a one-hot vector that indicate which word in the vocab is present in a specific location in the title, and $\hat{y}$ is the softmax distribtuion over all words in the vocab.

Another important evaluation metric is Bilingual Evaluation Understudy (BLEU), which is a score of how similar two sentences are (Papineni et al., 2002).

4.4 Model hyperparameters

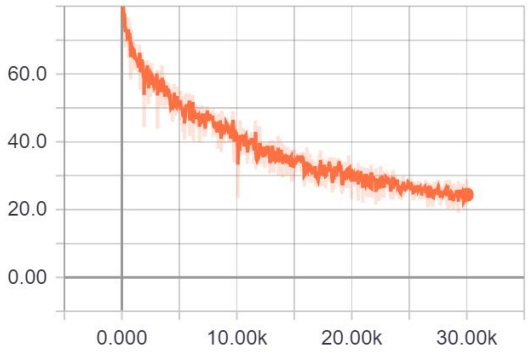
Our model and training are specified by multiple hyperparameters:

- Max input length: The max length of each article passed to the model. The average article length in the data was around 340 words, but to speed up the initial training we started with 150 words as max input length.
- Max output length: The max length of each headline generated by the model. The average headline length in the data was around 10 words, which is what we started with.
- Batch size: The size of each batch in the batched stochastic gradient descent, we started with 32 as batch size.
- Vocab size: The vocab size is critical to the success of the model. We started with 100000 but ran into memory issues, so we reduced it to 15000.
- Dropout rate: The probability to drop/keep links in cells while training the weights. We used a value of 0.2.

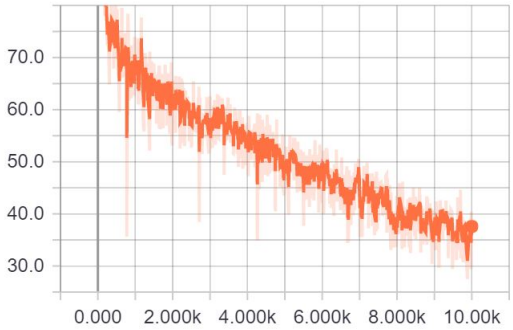
The method we followed in training the model to avoid overfitting was evaluating the model on the validation set after a certain number of batches. Then we save the model that had the best BLEU score. This way assures

that when the model overfits the training data, we have the model that have better generalization saved.

5 Results

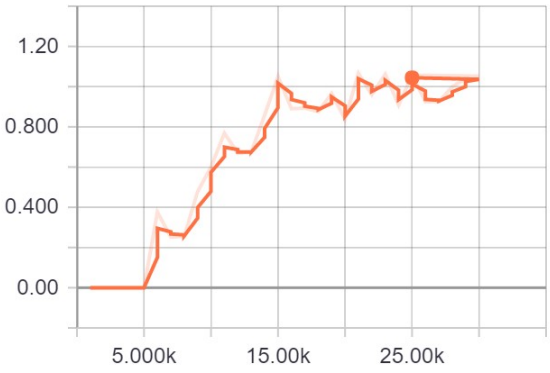


(a) Standard model

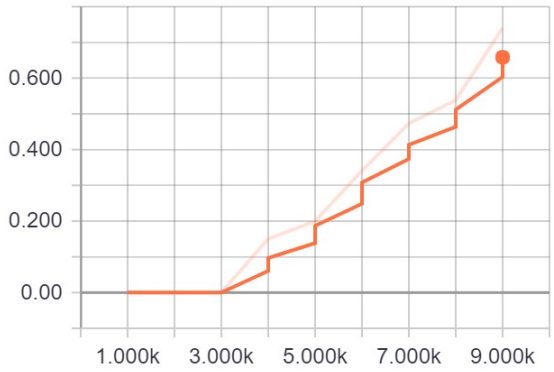


(b) Attention model

Figure 4: Batched trained vs. loss for both standard and attention models



(a) Standard model



(b) Attention model

Figure 5: Batched trained vs. BLUE score for both standard and attention models

5.1 Numerical Metrics

We used Tensorflow and the Luong et al. (2017) tutorial to implement the standard seq2seq and the seq2seq with attention models. We trained the standard model for 30,000 steps, where each step is a full batch. This covers about 60 epochs. On the other hand, due to limitations in computing resources, we were only able to train the attention model for 10,000 steps, which is about 20 epochs. Figure 4 shows the training loss for the standard and the attention models respectively as a function of the number of train steps. As can be seen in Table 2, the standard model performs better than the attention model in the BLEU metric. However, looking at Figure 5, which shows the BLEU scores for the two models on the test dataset, we can see that at 10K steps, the attention performed better, and so showing a higher potential.

5.2 Observations

Looking at samples of headlines generated by the standard and the attention models, we find some general themes. We find that generated headlines are usually very related to the article's topic. However, we also find that it tends to make up the details mentioned in the article. Finally, neither of the models seem considerably better than the other. In the first example given in Table 3, both of the models talk about the car accident. But, they also made up the numbers and the location of the accident. The standard model also decided all of these people are related and the accident was, in fact, a flip over, which both are details that weren't mentioned in the original article.

Model	Dev		Test	
	Perplexity	BLEU	Perplexity	BLEU
Standard (30K)	3969	0.9697	3663	1.053
Standard (10K)	1570	0.6354	1512	0.6003
Attention (10K)	2700	0.6653	2417	0.7408

Table 2: Models numeric results

Model	Generated headline
Human	6 Died and Injured in a Traffic Accident East of Quwaieyah.
Standard	8 of One Family Died and Injured in a Car Flip Over on Khurais.
Attention	Three Died and Seven Injured in a Traffic Accident to Arar.
Human	Meteorology authority forecasts dust storms in the northeastern and central regions of the Kingdom.
Standard	Forecasts of thunderstorms in Asir and Albaha and Jazan and Najran.
Attention	Meteorology authority forecasts thunder weather in regions regions regions of the the kingdom in to west.
Human	The crown prince heads a meeting with security officials.
Standard	Governor of Najran <UNK> <UNK> <UNK>.
Attention	The king receives the governor of Makkah.

Table 3: Translated examples of headlines

6 Future Work

As we saw in the previous section, despite having been trained less, the attention model showed more promise at 10K train steps. Therefore, the intuitive next thing to try is to train the attention model for more epochs. Moreover, one interesting direction for this project is to add a copying mechanism as in Gu et al. (2016). The copying mechanism may help better incorporate the details of the article (numbers, nouns, etc.) in the title as well as help to avoid predicting UNK as a part of the headlines.

7 Contributions

All team members have contributted equally to all parts of this project.

References

M. Alhagri. Saudi newspapers arabic corpus (saudinewsnet), 2015. URL `http://github.com/ParallelMazen/SaudiNewsNet`.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. Incorporating copying mechanism in sequence-to-sequence learning. *CoRR*, abs/1603.06393, 2016. URL `http://arxiv.org/abs/1603.06393`.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8): 1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `http://dx.doi.org/10.1162/neco.1997.9.8.1735`.

Konstantin Lopyrev. Generating news headlines with recurrent neural networks. *CoRR*, abs/1512.01712, 2015. URL `http://arxiv.org/abs/1512.01712`.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015. URL `http://arxiv.org/abs/1508.04025`.

Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. Neural machine translation (seq2seq) tutorial. 2017. URL `https://github.com/tensorflow/nmt`.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. Bleu: a method for automatic evaluation of machine translation. pp. 311–318, 2002.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014. URL `http://arxiv.org/abs/1409.3215`.