News2Headline

Recurrent neural networks for headline generation

Alberto Mario Bellini
University of Illinois at Chicago
Politecnico di Milano
abelli6@uic.edu

Flavio di Palo
University of Illinois at Chicago
Politecnico di Milano
fdipal2@uic.edu

Abstract

Automatic headline generation is a sub-task of document summarization with many reported applications. The aim of this paper is to describe the approach that we followed to predict article headlines starting from their content. We first introduce the reasons and related work that inspired us, illustrate the dataset we used, present some of the fundamental preprocessing steps performed on the dataset and then we dive into the details of the model we used. We are proposing two different models, both made up by an Encoder-Decoder neural network architecture that exploits pre-trained GloVe embeddings. In the last section we present results of our models against the state-of-the-art techniques using three different evaluation metrics: BLEU, semantic similarity and syntactic correctness. This work has been developed as a class project for UIC CS 521: Statistical Natural Language Processing.

1 Introduction

In this work we are addressing the problem of automatic headline generation from news articles.
We think this problem is relevant since, with the diffusion of social networks, more and more online articles are shared everyday. Sometimes article headlines are crafted to be sensational in order to push people to share it without even reading the full article. We think that automatic methods that are able to identify the article content and generate appropriate headlines could be helpful in this context.
In addition to this reason, we are curious to know to what extent a machine is able to extract the meaning of an article to predict a meaningful headline.
In this work we make use of a *seq2seq* model (Sutskever et al., 2014), which is widely used in machine translation, and adapt it in order to predict headlines starting from article news. We propose two different version of the same model: the first produces the headline by considering the most probable word given the previously predicted words and the second is built to learn an embedding of the most appropriate next headline word given the previous ones. We evaluate our results using BLEU, semantic similarity and syntactic correctness. Our models are compared with a baseline based on TF-IDF technique.

2 Related Work

The work from which we took major inspiration for this project is (Lopyrev, 2015) where the author describes an application of an encoder-decoder recurrent neural network with LSTM units (Hochreiter and Schmidhuber, 1997) and attention mechanism with the objective of generating headlines from the text of news articles. The proposed model is quite effective at concisely paraphrasing news articles.The BLEU achieved by this approach will be compared with our results in the results section.
It is essentially a multilayered Long Short-Term Memory (LSTM) model used to map the input sequence to a vector of a fixed dimension, and then another deep LSTM decodes the target sequence from the vector. the fixed dimension vector represent the encoder hidden state. In this particular work an English-French automatic translation task is addressed as an example for the *seq2seq* model. The task we are considering in this work is similar under certain circumstances since it can be considered as a mapping of an article as an input sequence in the ”article” language to the output sequence in the ”headline” language.
In (Padmankumar and Saran, 2016) the authors are proposing a method to perform unsupervised extractive and abstractive text summarization using sentence embeddings; this work pushed us to explore the possibility to exploit pre-trained GloVe embeddings coupled with our neural network architecture.

We also considered (Takase et al., 2016) in which the authors propose a fully data-driven approach to abstractive sentence summarization. Their method utilizes a local attention-based model that generates each word of the summary conditioned on the input sentence. We initially took inspiration from this work. The main contribution of it to our research is to make us consider the attention mechanism discussed within the paper as first extension we would like to add in our future work, since at the moment our model lacks this mechanism.

3 Dataset

The corpus that we decided to use was taken from the "All the news" dataset publicly available on Kaggle and collected by Andrew Thompson. It consists of 142.570 news taken from 15 different American publications which are: The New York Times, Breitbart, CNN, Business Insider, The Atlantic, Fox News, Talking Points Memo, Buzzfeed News, National Review, New York Post, The Guardian, NPR, Reuters, Vox and The Washington Post.

The data primarily falls between the years of 2016 and July 2017, although there is a not-insignificant number of articles from 2015.
Out of the 8 different attributes that were provided for each article, we kept only the headline and the article body because information such as date of publication and year were not relevant to the extent of our work.

3.1 Preprocessing

The first stage of data preprocessing consisted in removing all recurrent sentences in headlines. This because many of them were containing phrases such as " - The New York Times" that were repeating in each news from that publication. Without performing this step our model would have been significantly biased towards generating headlines with this recurrent sentences.
The second stage consisted in removing all bad characters, such non-printable Unicode words, which were considered as noise. For this reason, instead of using the whole Unicode character space, we decided to use ASCII characters only and removed everything else.
In the third stage we focused our attention into filtering out all those news which were too short to be taken into account. We first set two thresholds for the minimum headline length and article length respectively to 5 and 10 words.
We filtered out all those news whose headline or article body was shorter than the the two thresholds. All words were then transformed to their lowercase representation to help the tokenization process and to work better with the pre-trained GloVe embeddings that will be described later.
In the fourth stage, we computed *Term Frequency-Inverse Document Frequency (TF-IDF)* on the whole dataset preprocessed until this stage and retrieved the the 10.000, 30.000, 50.000 and 70.000 most important words. During the process we used a smoothed inverse document frequency to penalize the most common words. The goal of this step was to find a restricted, but meaningful, vocabulary of words to use during the training phase. In order to reduce the computational time need to train the following models, we decided to use the 50.000 most important words computed using TF-IDF and replaced all the other words with an [UNKNOWN] token.
During the fifth stage of preprocessing we started to prepare the inputs for our model. The model receives as input fixed length vectors of size 20 and 30 tokens respectively for the headlines and the articles. This size is limited due to the enormous processing power required to deal with longer sequences. We first truncated all the headlines and articles to the maximum input size of 20 and 30 words. Then we tokenized all the words using the NLTK Framework.
After the tokenization process the size of the created lists of tokens were, eventually, still greater than the specified maximum size due to the fact that punctuation was kept and considered as individual tokens. Moreover we wanted to keep, for each headline and article, only the first paragraph so in the sixth step we truncated the headlines and articles again. This time however we endorsed the following strategy: each headline or article were truncated either to the first stop word among ".", "!" and "?" or, in case no stop words were found in this range, to the maximum length described above. During this process many articles and headlines became too short, since the first period was occurring too early.
To avoid dealing with short news, in stage seven, we filtered out again all those samples that were under the minimum length after the previous preprocessing steps.
We then added [START] and [STOP] tokens to

#	Stage	# News	Avg. HL	Min HL	Max HL	Avg. AL	Min AL	Max AL
0	Unprocessed	142570	10.5	0.0	33.0	741.4	0	50035.0
1	Hadline filtering	142570	10.0	0.0	33.0	741.4	0	50035.0
2	ASCII Filtering	142570	9.8	0.0	32.0	733.9	0	50015.0
3	Small news filtering	137935	10.0	5.0	32	731.4	10.0	50015.0
5	Maxlen truncate	137935	10.0	5.0	20.0	29.9	10.0	30.0
5	Tokenization	137935	10.7	5.0	32.0	32.6	11	49.0
6	First paragraph truncate	137935	10.3	1.0	20.0	23.7	0.0	30.0
7	Small news filtering	126330	10.4	5.0	20.0	25.1	10.0	30.0
	Fully processed	126330	10.4	5.0	20.0	25.1	10.0	30.0

Table 1: Dataset evolution, from unprocessed to fully processed. We show, for each stage, the number of available news, the average, minimum and maximum headline length, and the average, minimum and maximum article length

our headlines and articles and padding them to the fixed length of 20 and 30 tokens, respectively, using a specific [PADDING] token.

3.2 Embeddings

Our model works with GloVe embeddings (Jeffrey Pennington, 2014) downloaded from the Stanford website.
This embeddings have been pre-trained on Wikipedia pages and Gigaword 5 dataset. They consist of a total of 400.000 different embeddings. Due to computational issues, we decided to use 50-dimension embeddings, the most compressed format available.
The first layer of our model, that will be described in detail later, has an embedding layer that maps the input with the corresponding embedding which, in turn, is then fed into the LSTM. For this reason we needed to map our tokenized and padded input vectors to their corresponding embedding. Out of the 400.000 available embeddings we computed a smaller embedding matrix that was based only of the 50.000 most important words found with TF-IDF. Out of the words returned by TF-IDF we kept only the ones that had a corresponding embedding among the ones downloaded. In the end, out of the 50.000 words returned by TF-IDF, only 47625 were used and the remaining words were set to the [UNKNOWN] token. At this point each headline and article were mapped to their relative embedding index, ready to be fed into the model. The first layer of the model will later map this index to the corresponding embedding representing that word. Table 1 shows how the dataset evolves throughout the various preprocessing stages.

3.3 Custom tokens

As mentioned, we used four different tokens in total to fully prepare the input for our model. These tokens are:

- [START] : Indicates the beginning of a headline or article.
- [STOP] : Indicates the end of a headline or article.
- [UNKNOWN] : Indicates an Out Of Vocabulary (OOV) word.
- [PADDING] : Used to pad sequences to specific lengths.

These tokens needed to have an embedded representation too, and we decided to assign them the embeddings of four words outside of our vocabulary.
During the preprocessing phase described earlier we inserted the [START] token at the beginning of each headline and article in order to have the model understanding that this specific token was representing the beginning of a sentence. The [STOP] token was added at the end of the sentence according to a specific strategy. If the sentence was shorter than the input size, the token was inserted right after the end of the sentence, and the remaining space was filled with the [PADDING] token. Otherwise, if the sentence was longer than the input size, as mentioned above, it was first truncated to the desired length and then the last word was overridden with the [STOP] token. In the end, all articles and headlines are fed in the model in a representation which always consists of a [START] and a [STOP] token. Padded sentences

always contain one or more [PADDING] tokens, depending on the number of free slots that needed to be filled to reach the input length and all OOV words are replaced with the [UNKNOWN] token.

4 Model

We decided to address the problem of generating meaningful headlines using a model which recalls the structure of *seq2seq*. The architecture is composed by two high level components: one encoder and one decoder. Both components are made up by one LSTM cell each with latent dimension set to 256. The two cells collaborate to predict headlines.

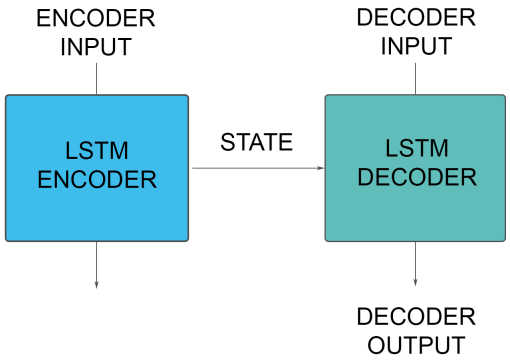

Figure 1: Encoder-Decoder architecture

The goal of the encoder is to create an encoded representation of the articles it is fed with. During training it understands how to summarize articles and how to capture their semantics. We do not consider the encoder outputs but we keep its internal hidden and memory states which represent a certain article semantics.
We use these states as initial ones for the decoder which, during training, is fed with the headline corresponding to the article that was given to the encoder. We train our decoder, given these two inputs, to predict the same headline shifted in time by one time instant. In other words we are doing what is commonly known as *Teacher Forcing* (Lamb et al., 2016), a method to train recurrent neural network models efficiently using the output from a prior time step as input. Given the encoded representation of the article and the [START] token, the decoder is trained to predict the first word in the original headline. Concurrently, given the first word and the same encoded representation of the article, the decoder is trained to predict the second word in the original headline and so on for all the words in the sentence until the [STOP] token is reached. This strategy is widely used to develop deep learning language models used in machine translation, text summarization and image captioning.
In order to feed our LSTMs the embedded words present in the headlines and articles, we added to both units an *embedding layer* whose weights were set to the pre-trained GloVe embeddings matrix computed during preprocessing. As described earlier, in the last stage of preprocessing, each sentence is mapped to its corresponding GloVe index. The goal of this layer is to translate that index to the corresponding GloVe embedding vector, in order to provide the LSTM with the semantic representation of the words in the emdedded hyperspace.
We set the *embedding layer* to be non trainable because we didn't to modify the embeddings but keep the pre-trained ones. We have developed two different models which share the same architecture described above for what concerns input layers and encoder-decoder structure but differ on how they produce their outputs.

4.1 Probabilistic output approach

The first model that we came up with has as output a dense layer with a number of neurons equal to the amount of words in our vocabulary, each with a *sigmoid* activation function. A *softmax* is then applied over the whole output space in order to distribute probabilities to each word. In this way, for each input token, the decoder will output over all words in the vocabulary the probability that each one should follow the current token. This approach is really powerful, because give us control over the predicted headlines. When the [STOP] token is predicted as most probable following word, indicating that the prediction is over, we can decide to enlarge the predicted headline just by taking as next word the second most probable one. However this approach has the drawback that is extremely computationally expensive since during training we need to provide as decoder target a very sparse vector.
We used the *categorical crossentropy* loss function and trained the model for 100 epochs using early stopping to avoid overfitting. *Stochastic gradient descent (SGD)* with *Root Mean Square Propagation (RMSProp)* was used as optimizer in order to minimize the loss and adapt the learning rate during the training phase.

Real headline	Predicted headline	BLEU score
Still puzzled by the election	The election in 2016 race	0.10
Syrian planes strike near Damascus	Syria strike	0.05
For Pope Francis year of reconciliation	Pope Francis	0.10

Table 2: Some predicted headlines using the probabilistic approach

4.2 Embedded output approach

The second model shares the same core architecture of the first one but differs on how it produces its outputs. Instead of distributing probabilities over words it produces directly an embedding whose size is the same one of the embeddings we use as inputs, set to 50. In order to produce actual embeddings, the output is a dense layer whose size is set to 50 neurons.
We then train the model using *cosine proximity* as loss function in order to maximize similarity between the produced embedding and the correct one. Moreover, since the embedding space is continuous and not restricted to $(-1, +1)$, we couldn't use anymore *sigmoid* activation functions because otherwise we would have had vectors values limited to that specific range. For this reason we picked *linear* activation functions to obtain embeddings whose values were ranging in the whole real space.
With this approach, target matrices are not as sparse as the ones used with the first model and thus we were able to use all the words present in our dataset without limiting them to the 50.000 most important ones returned by TF-IDF. Afterwards, in order to get the predicted word for a given token, we computed the cosine similarity between the output embedding and all the word embeddings in our vocabulary. The predicted word was defined as the closest one to the predicted embedding.

4.3 Motivations

We decided to follow this approach because, in order to generate meaningful headlines, we were interested in a model that was able to capture long term relationships among text sequences. LSTMs are extremely fit for this task since, using input, output and forget gates, can selectively let information flow in or out the memory cell. For this reason they are extremely flexible when it comes to learning long time dependencies between sentences seen during training. Moreover LSTMs can be fed with enormous amount of data thanks to this gated structure that prevents vanishing or exploding gradient issues common in standard RNNs. We discovered that our Encoder-Decoder architecture can learn related concepts even if these are spread across different articles and fed to the model in different time instants.

4.4 Changes with respect to literature

Common literature suggested us to follow first a probabilistic approach, in order to have more control over the headlines generated by the model. However what we think is new in our work is the approach in which we generate embeddings as predicted tokens and take as next predicted word the one whose embedding is closer to the one produced by the LSTM. In this way we are able exploit the power of the GloVe embeddings that we have and furthermore, thanks to the reduced output shape, we can even make our model more powerful and increase the latent dimension while keeping training times close to the probabilistic approach. We are even able to use 300-dimensional embeddings whereas the probabilistic model, due to its enormous complexity, can handle at most 50-dimensional vectors.

4.5 Inference

While many Deep Learning models are able to make predictions as soon as the model has been trained, our model acts differently and needs to be split while making inference. This because, while training, we feed the encoder and the decoder articles and headlines, respectively. We know both inputs and we easily compute the target output for the decoder by shifting in time the headline by one time step.
However, when making inference, we don't know the full headline because we want to predict it. For this reason we need to tear apart the encoder and the decoder and work on them separately. We first take the pre-processed input representation for the article and feed it to the encoder. Afterwards we take its internal *hidden* and *memory* states and manually set them as initial states for the decoder.

This procedure needs to be repeated for each article we want to predict the headline for. Then we prepare the inputs for our decoder as a vector which contains all [PADDING] tokens except for the first one which is set to the [START] token. At this point we predict the full output of the LSTM which, since is of fixed size, will consist of 20 different values. We take as first predicted word the first one of these values, we save it and then we append it to the input sequence, after the [START] token. This process goes on until the next predicted word is the [STOP] token which indicates that the full headline has been predicted completely, according to the model. Depending on the approach used, the predicted word is obtained either as the most probable one or the closest one to the produced embedding.

5 Evaluation

We evaluated our results with three different metrics. The first one is the BLEU, a metric used to evaluate the amount of words that the predicted headline has in common with real headline.
The second metric used is semantic similarity computed on the title embeddings. It is used to understand how much our model is able to predict headlines that are semantically close to the real headlines. The last metric used considers the syntactic correctness of the produced headline.

5.1 BLEU

The BLEU (Papineni, 2002) is a metric widely used in machine translation domains. BLEU uses a modified form of precision to compare a candidate translation against multiple reference translations. Machine translation systems have been known to over generate reasonable words in predicted sentences, resulting in improbable, but high-precision, translations. BLEU is computed starting from a new precision metric that is called modified n-gram precision. The Modified n-gram Precision, exploits the simple concept that a reference word should be considered exhausted after a matching candidate word is identified. The complete BLEU implementation can be found on(Papineni, 2002)

5.2 Embedding Similarity

In order to have a measure of how much the real and predicted headline are similar from the semantic standpoint we are computing the cosine similarity between the real headline and the predicted headline embeddings. In order to obtain an embedding for a sentence we average the GloVe embeddings of each word contained in the sentence. Once we have computed the reference headline embedding and the candidate headline embedding, we have two n-dimensional vectors, where n is the size of the GloVe embedding used for the procedure. The embedding similarity is computed by computing the cosine Similarity of this two vectors.

5.3 Headline Syntax

Seeing our model results we noticed that the generated headlines often are syntactically correct. In order to have a more precise measure from this intuition we are using a English parser available in NLTK library that is based on The Penn Treebank dataset in order to evaluate the percentage of syntactically correct titles generated by each approach.

6 Baseline

In order to understand how our model performs compared to other methods we developed a simple baseline. The baseline consists in predicting as headline the first 5 words of the the article as it is after the preprocessing phase. This means that the title will be composed of the first 5 words present in the article after TF-IDF pre-processing.

7 Results

In this section we report some of our most important results. The results have been computed performing 3-fold cross validation on the complete dataset. Table 3 presents, for each of the algorithms discussed above, the result achieved on each evaluation metric considering the average achieved over the 3 different folds and the standard deviation among the various results.
The best BLEU performance is achieved by (Lopyrev, 2015) model. It is explainable by the fact that the reference work extends the basic architecture we have presented with 2 different attention mechanisms.
Surprisingly we notice that our TF-IDF approach is better than both our *seq2seq* models for what concerns BLEU score. This is why the *seq2seq* model are generative models working on embeddings, while TF-IDF is considering as article headlines some of the most relevant words in the arti-

Model	Avg B.	Std B.	Avg. ES.	Std ES.	Avg SC.	Std SC.
Model(Lopyrev, 2015)	0.091	n.a	n.a	n.a	n.a	n.a
Probabilistic Model	0.058	0.0017	0.779	0.0085	0.2891	0.0320
TF-IDF	0.080	0.0018	0.661	0.0042	0.0545	0.0043
Embedding Model	0.019	0.0015	0.619	0.0079	0.2343	0.0289

Table 3: results evaluated with 3-fold cross validation, Avg and Std represents the average and the standard deviation of the results obtained on the 3 folds. B. stands for BLEU, ES. stands for embedding similarity and SC. stands for syntax correctness

cle. For this reason the TF-IDF approach predicts on average more word that are present in the actual headline.
We can state that for what concerns embedding similarity the probabilistic model achieves better performances than TF-IDF. This is understandable since our probabilistic model works using GloVe embeddings and it has, on average, a better understanding of the article semantics compared to the TF-IDF approach.
For what concerns embedding similarity we don't have a measure for the approach attempted in (Lopyrev, 2015) since those measure is not provided in the paper.

From our tests we can also state that our embedding model has not comparable performances with respect to the probabilistic model and this is why the task of producing a complete new embedding is very difficult. This idea seems to be not effective in this particular context.
For what concerns syntactic correctness we derive from our results that both our models are performing better than the TF-IDF approach; it is understandable since our models are based on a *seq2seq* architecture that is able to learn the structure of the headline. Our TF-IDF baseline is basically sampling 5 words in the article and for this reason it is very difficult to produce a syntactically correct headline.

We considered statistical significance for all the results that are in the same range. Starting from results shown in Table 3 we performed a two tailed hypothesis test in order to understand if the difference between the BLEU achieved by the TF-IDF model and the embedding model is statistically significant or not. We have used GraphPad online tool to perform the test.
We can conclude that the two-tailed p-value equals 0.0002 by conventional criteria, this difference is considered to be extremely statistically significant.
We evaluated also the embedding similarity achieved by the probabilistic model vs the embedding similarity obtained by the TF-IDF approach; in this case the p-value obtained by the two-tailed test is equal to 0.0001 and that means that the difference in the two means is statistically significant.

8 Conclusions and Future Work

In this work we are considering an Encoder-Decoder neural network architecture in order to solve the headline generation problem. We have demonstrated the effectiveness of our approach in the context of headline generation and compared it to both the state-of-the-art model and our TF-IDF based approach.
In the future we plan to explore different models that are computationally lighter. This kind of models would give us the possibility to explore and use a bigger portion of the input article to predict more meaningful headlines. In parallel we plan to make improvements to our best model by introducing an attention mechanism and using bi-directional LTSMs.

9 Appendix

9.1 Contributions

We both studied the available literature and come up with different ideas about the model before starting to write code. After a discussion phase we started to implement the project. During the implementation phase we split the tasks in the following way:

Alberto Mario Bellini:

- Data preprocessing and exploration
- Probabilistic model development
- Embedding model development

Flavio Di Palo:

- Data loading with Keras generators
- Evaluation Metrics Development
- TF-IDF approach development
- Statistical Significance on Results

9.2 Github Repository

All the code we have developed for this project is available on GitHub.

9.3 Consideration on Computing Power Nedded

Due to the complexity of the model we exploited Google Cloud Services to setup a VM instance with 96GB or RAM and one NVIDIA Tesla P100 with 16GB of HBMO memory

References

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Christopher D. Manning Jeffrey Pennington, Richard Socher. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4601–4609. Curran Associates, Inc.

Konstantin Lopyrev. 2015. Generating news headlines with recurrent neural networks. *CoRR*, abs/1512.01712.

Aishwarya Padmankumar and Akankasha Saran. 2016. Unsupervised text summarization using sentence embeddings. *CoRR*.

Toddward Weijing Papineni, Roukos. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1054–1059. Association for Computational Linguistics.