Title Generation

Aditi Singh, Ishita Agarwal, Kaamraan Khan, Vedant Desai

1 Problem Definition

Headline generation while preparing to write about any topic is very important as headings gain readers attention and based on that most of the users decide whether they want to read a particular article. If the writer fails to give a relevant headline to a good article the user may not even consider reading it. The aim of the project is to develop a generalized architecture which generates heading for the given text, which can be an article, a book's paragraph, news paper contents or any legal document. A model is proposed which makes use of encoder – decoder recurrent neural network architecture consisting of LSTM and attention units.

2 Background

Word embedding is a technique used for representing words. It allows words used in similar manner to have similar representation. In this each word is represented as a real valued vector of d dimensions in already defined vector space. In our model we are using Google's pre-trained word2vec file to map words present in our dataset to vectors. It is also used for weight initialization. So, we aim to utilize this trait by representing word in word embedding feature vector form.

RNN (Recurrent neural networks) are widely used for converting text from one form to other. These models are trained on large set of input and output data. After training when new input data is provided the model will generate expected output data. These models are also capable of recalling facts and statements related to a text provided as input. So, we aim to use an encoder-decoder architecture, both of which will make use of RNN.

A type of recurrent neural network is Long Short-Term Memory (LSTM) which can learn order dependencies in problems involving sequence predictions. Both encoder and decoder make use of LSTM. Encoder uses a Bi-directional LSTM layer which helps in extraction of information from given text. It reads one word at a time and updates the hidden layer based on words it has already read and the current word. Decoder generates summary of one word at a time by making use of a uni-directional LSTM layer. To generate probability distribution of next word the decoder makes use of information it has written before as well as information from encoder.

Beam Search is one of the most popular search techniques for Deep natural language processing algorithms, wherein we divide the text into sequences and we select the best alternatives based on the beamwidth. It selects n=beam width highly probable alternatives for a sequence given. Beamwidth should be chosen carefully otherwise it may take a lot of computational power and memory, though more the alternatives better the results. So instead of having just 1 possible headline, the model will select more than 1 headline and will use some of the best to get the results.

Mechanisms like RNN and LSTM are not enough to generate efficient titles that's why to increase the quality of our predictions we have made use of the attention mechanism in our last layer. Attention mechanism helps in selecting the most appropriate word to which attention needs to be given. It helps the model to remember important aspects of the input in an efficient manner. LSTM overcomes the limitation of RNN which requires us to send fixed length input. Attention mechanism helps the model to focus more on important features based on the generated output. The attention model consists of an encoder and decoder. The encoder is LSTM based and is used to process the input given by converting the whole sentence to a context vector. This makes the last hidden state of LSTM which is expected to return good predictions for the given input sentence. The decoder then produces words one after the other to make a comprehensible headline. The traditional models such as RNN or LSTM which make use of encoder-decoder models generally discard all the intermediate states of the encoder and initialize the decoder only by making use of the encoder's final state. This mechanism can work when the sequences are small but as the length of the sequence increases making a single vector becomes cumbersome and the performance of the model decreases significantly. Attention mechanism overcomes this limitation by utilizing all the intermediate states in construction of the context vector which generally tells the weight i.e. attention which the models need to give to each word. The encoder decoder model which makes use of attention gives better performance because it uses an interface of context vector which is given by using attention mechanism to know which words the model needs to pay attention to.

3 Data-Set Used

We have tried to use data set from diverse domains to make a generic architecture. We have used data set available on BBC news website which consists of 2225 documents from the BBC news website corresponding to stories in five topical areas. It also consists of 737 documents from the BBC Sport website corresponding to sports news articles in five topical areas.

We have also scraped data from medium which consists of summary of the articles present on the medium website together with its title. This dataset consists of 1700 rows where each row consists of the article summary and title.

Department of justice press releases dataset is used which is scraped from data.gov site of USA using beautiful soup. It consists of 13,087 press releases but does not includes data which have been labelled as speeches.

All news data set contains articles from publications such as CNN, FOX News, New York Times, RSS etc. The data was scraped using beautiful soup from archive.org and was stored in SQLite, which was stored in csv of size 50,000.

We have done pre-processing on the content as well as the headings. These include: converting everything to lowercase, removing HTML tags, contraction mapping, removing "", removing any text inside the parenthesis, eliminating punctuation and special characters, removing stop words and short words.

In total our dataset when combined contains 1 lakh plus data points or rows where each row consists of the text or the content using which headings will be predicted and the actual heading of the given text or content.

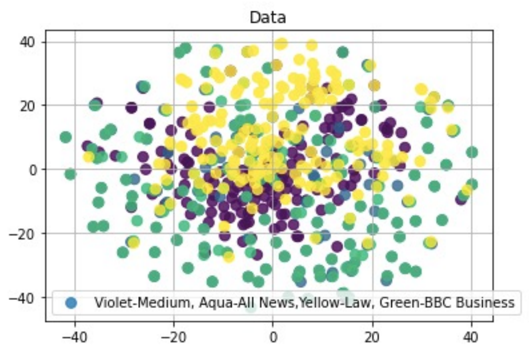


Figure 1: Scatter plot showing most frequently occurring words accross various datasets

We have tried to analyze the dataset by visualizing it with the help of a scatter plot. For this we have taken 200 most frequently occurring words in the heading of each dataset. As we have used four dataset spread accross various domains we got a total of 800 most frequently words in headings accross all the datasets. From the scatter plot it can be seen that most of the words which occur frequently are similar as most of the words are present close to each other. We can also see that our dataset is very generic as not all the most frequently occurring words are present in close proximity to each other some are scattered away from the centre.By using dataset from 4 different domain we were able to generalize our model to predict headings from different domains efficiently.

4 Proposed Solution Sketch

After applying the necessary preprocessing as described above we need to change the words into a form that can be passed to our model. We can do 2 things either assign a number corresponding to each word (make a word2idx dictionary) or use a vector representation i.e. Google word2vec to create a word embedding of a large corpus of data by producing a vector space of several hundred dimensions, in our case we have kept the dimensions to 300. Each unique word in the corpus is assigned a vector. It is a more meaningful representation as words that have similar or close context are assigned vector spaces that are close to each other in space.

The vector or the word embeddings so formed are passed into the encoder network i.e. has LSTM layers in it and provide an output feature vector that holds the necessary information to map the input to output. We are passing the hidden layer and output layer to the decoder which is similar architecture i.e. LSTM and gives the output that is closest to the actual input i.e. headlines in our case. As we have a lot of articles simultaneously being passed to the network we add LSTM for identifying the useful information and prevents the generation of ambiguous headlines. LSTM keeps the mapped representation of every article, therefore, enabling a hierarchical structure for the decoder.

The decoder uses the obtained representations from all the models i.e. it is able to access the hidden layers of the article. The next step is to convert the vector to words, so the decoder takes the words of the document summary and forms a context vector ct.

Next we have applied the attention network after the encoder-decoder model as have been already described, it gives more weightage to the background words. The attention model is used with an encoder-decoder architecture. The input it receives is the encoder states(hidden and output) as well as the current decoder state. IT gives the output as floating-point numbers which represents the weights of the model.

We have trained the model on various datasets having varying sample sizes and have used them to train our model. For Encoder-We have kept a batch size of 4, hidden dimension=5, learning rate=0.001, the optimizer is adam, batch size is 4 ran the code for 200 epochs. We have integrated a dropout layer in LSTM with a dropout probability of 0.1. Similar parameters are being used for the decoder. For loss calculation we are using L1 loss(also called Least Absolute Deviations) as it creates a criteria to calculate mean absolute error between the corresponding elements in the input as well as the output. It is the summation of error between the actual and predicted value and have to be minimized to get the predicted headings close to the actual ones.

Thereafter we are applying a fully connected layer to get the final results.

Next we will talk about the evaluation metrics we have used for our output:

- Bleu score The Bilingual Evaluation Understudy Score or BLEU score looks at the fraction of different lengths n-grams of the expected heading which are given as output by the model. It gives a score by comparing consecutive phrases of actual result with that of generated result. It counts the matches found in a weighted manner. The matches are position independent and higher score means that high similarity is there between actual and predicted outcomes.

- Google word2vec distance similarity: The distance similarity metric suggests that the distances between the embedded vectors of words are semantically meaningful. The distance between text document P and Q is calculated by measuring the minimum cumulative distance required by words of text document P to match the vector of document Q. This metric measures the dissimilarity between two documents by calculating the distance one document has to travel to reach the word embedding of another document. Here, the distance similarity is measured between actual headings and the given headings.

- Cosine similarity: It measures the similarity between two non zero vectors by measuring the cosine angle between them. In our case we are finding cosine similarity between the actual headings and the predicted ones to see how similar are our predictions to that of the ground truth i.e. actual headings.

The above described architecture can be compressed into a diagrammatic representation shown below.

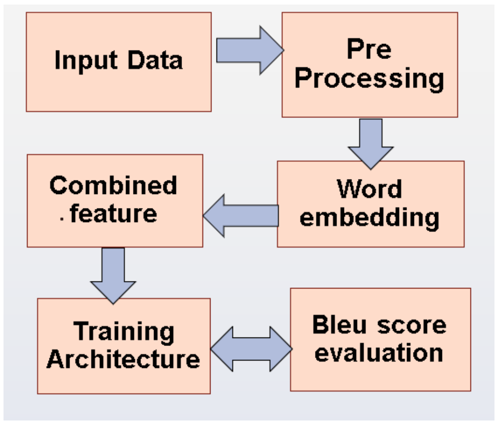


Figure 2: Proposed architecture

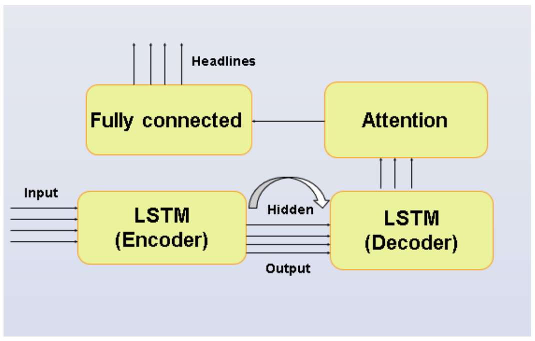


Figure 3: Training Architecture Explained

5 Literature Review

In the paper “Generating News Headlines with Recurrent Neural Networks”, an encoder-decoder model of RNN with LSTM unit has been described. Performance of the model was measured by evaluating training and holdout loss and by using BLEU evaluation metric which looks at fraction of different lengths n-grams of the expected heading which are given as output by the model. It was observed that setup training loss was higher than holdout loss. The model was very effective while predicting the headlines when newspaper fed was same as those it was trained on but, it gave mixed performance when articles from papers it was not trained on were given. Therefore, we are using a diverse data set to forgo this issue. The model can also be improved by making use of a bi-directional RNN.

The paper “Automated title generation in English language using NLP”, describes an approach for generating title without making use of the whole document. It takes an article as input and gives a title as output by using different approaches such as frequency prioritizing, combination of noun and adjective or title based on idiom. The system performance is dependent on the size of the document. In some cases, the output generated was not very close to the expected output. Therefore, we observed that the model of first paper performs better as compared to the model of second one.

In the paper “A neural attention model for abstractive sentence summarization. In: Empirical Methods in Natural Language Processing” the authors have applied the local attention model to generate words for the summary based on the input fed to the model. The language model used is a feed-forward neural network language model that acts as a probabilistic model and they have used an encoder for conditional summarization. They have used 3 variants of encoders namely Bag-of-Words Encoder, convolutional encoder, and attention encoder. For generating headlines 2 decoders i.e. greedy and beam search decoders are used. The dataset used is DUC-2003 and DUC2004 and ROUGE-1 and ROUGE-2 as the evaluation metric.

In the paper ‘From neural sentence summarization to headline generation: a coarse-to-fine approach’. , the authors are using the encoder-decoder framework of the seq2seq model. The encoder converts the input text into a vector which will represent all the words and a decoder model which gives the output of the input. The decoder model used is RNN. Dataset used is news articles from the English Gigaword corpus. The evaluation metric used is ROGUE-1, ROGUE-2, and ROGUE-L.

In the paper “The Impact of Local Attention in LSTM for Abstractive Text Summarization”, the author proposed to use the local attention model together with LSTM. The dataset used in this research is taken from Amazon fine food reviews and for the purpose of evaluation GloVe dataset is being used, which brings in light that the global attention based model gives better ROUGE-1 score than the local one because it generates more pair of words which are actually present in the summary. The local attention model generates a better ROUGE-2 score than the global oneHere ROUGE-1 and ROUGE-2 are the evaluation metric on the basis of which performance of various models are being compared. The authors have mentioned that the accuracy can be improved by handling OOV while doing pre-processing and also, by generating optimal parameters. It can be observed that the model making use of attention mechanism gives better performance as compared to models just using LSTM, or RNN. This model even has scope of improvement which means that if we are able to overcome the shortcomings of the mentioned model we will be able to get better results with it.

6 Baselines

First, we have implemented the model described in ”Automated title generation in English language using NLP” which uses frequency prioritizing, POS tagging i.e. combination of noun and adjective or title based on phrases. Bleu score is shown in the diagram with the 1st bar graph.

The bleu score came out to be Next, we implemented a simple RNN model corresponding to which we will generate the bleu score. This model performs better than the previously described one as it was too naive and no learning was done. This model applying simple RNN formed the basis of our further applications where we apply advanced models keeping the implemented simple RNN model intact. , We are using a tensor flow library for simple RNN; it basically is a fully-connected RNN where we pass the output from the previous timestamp the next timestamp. It is helpful for our data because it only takes into consideration the current input but also doesn't forget about the previous elements. Because of this memory structure the entire context is used for making the predictions. In the diagram given the 2nd bar graph corresponds to bleu score of this model.

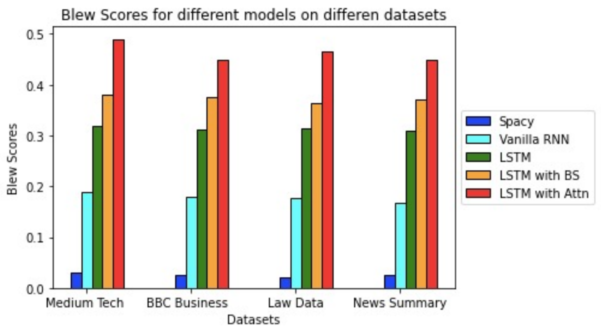
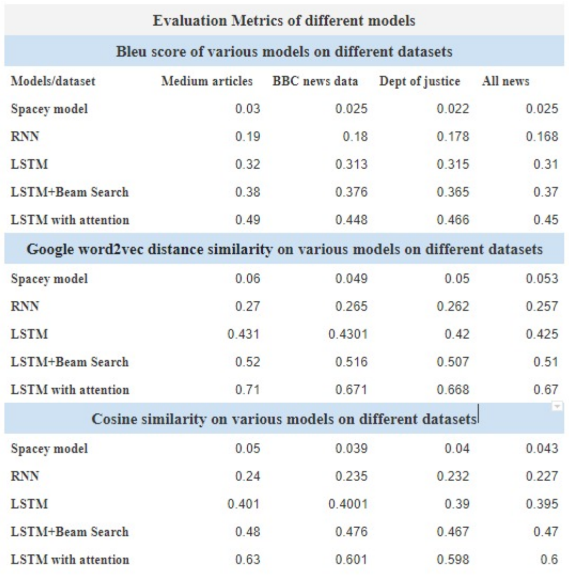


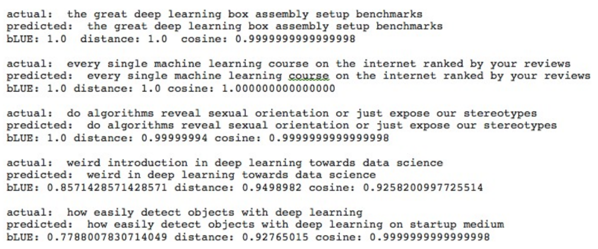
Figure 4: Graph showing BLEU score of different models accross various datasets

7 Results Produced



Evaluation Metrics of different models				
Bleu score of various models on different datasets				
Models/dataset	Medium articles	BBC news data	Dept of justice	All news
Spacey model	0.03	0.025	0.022	0.025
RNN	0.19	0.18	0.178	0.168
LSTM	0.32	0.313	0.315	0.31
LSTM+Beam Search	0.38	0.376	0.365	0.37
LSTM with attention	0.49	0.448	0.466	0.45
Google word2vec distance similarity on various models on different datasets				
Spacey model	0.06	0.049	0.05	0.053
RNN	0.27	0.265	0.262	0.257
LSTM	0.431	0.4301	0.42	0.425
LSTM+Beam Search	0.52	0.516	0.507	0.51
LSTM with attention	0.71	0.671	0.668	0.67
Cosine similarity on various models on different datasets				
Spacey model	0.05	0.039	0.04	0.043
RNN	0.24	0.235	0.232	0.227
LSTM	0.401	0.4001	0.39	0.395
LSTM+Beam Search	0.48	0.476	0.467	0.47
LSTM with attention	0.63	0.601	0.598	0.6

Figure 5: Table showing various evaluation metrics score for different models accross various datasets



```
actual:  the great deep learning box assembly setup benchmarks
predicted:  the great deep learning box assembly setup benchmarks
bLUE: 1.0  distance: 1.0  cosine: 0.9999999999999998

actual:  every single machine learning course on the internet ranked by your reviews
predicted:  every single machine learning course on the internet ranked by your reviews
bLUE: 1.0 distance: 1.0 cosine: 1.0000000000000000

actual:  do algorithms reveal sexual orientation or just expose our stereotypes
predicted:  do algorithms reveal sexual orientation or just expose our stereotypes
bLUE: 1.0 distance: 0.99999994 cosine: 0.9999999999999998

actual:  weird introduction in deep learning towards data science
predicted:  weird in deep learning towards data science
bLUE: 0.8571428571428571 distance: 0.9498982 cosine: 0.9258200997725514

actual:  how easily detect objects with deep learning
predicted:  how easily detect objects with deep learning on startup medium
bLUE: 0.7788007830714049 distance: 0.92765015 cosine: 0.9999999999999998
```

Figure 6: Headlines predicted by our proposed model

Figure 5 show the headline predicted by our proposed model together with the actual headline. It also gives details about various evaluation metrics such as BLEU, cosine similarity and distance similarity score.

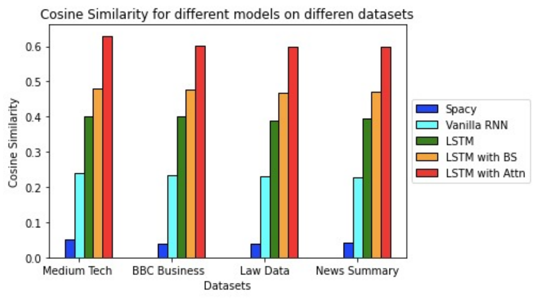


Figure 7: Graph showing cosine similarity of different models accross various datasets

From the above graph it can be seen that our proposed model has predicted titles having maximum similarity to the actual ones. The maximum cosine similarity is seen in LSTM using attention units and it gave maximum similarity accross various datasets.

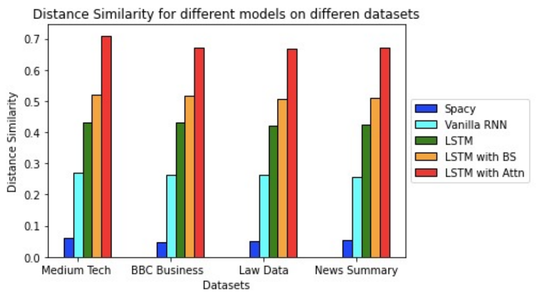


Figure 8: Graph showing distance similarity of different models accross various datasets

From the above graph it can be seen that our proposed model has predicted titles having maximum distance similarity to the actual ones. LSTM with attention model has given maximum distance similarity accross various datasets.

Hence, we can say that our proposed model making use of LSTM with attention unit has outperformed all the other above mentioned models and has predicted headings similar to the actual ones which can be seen from various graphs.

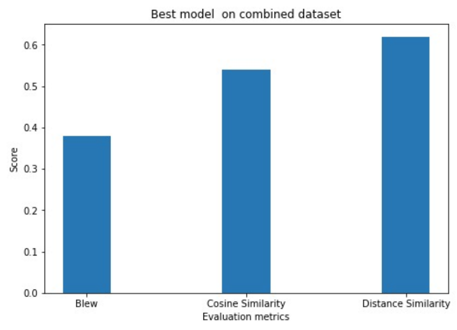


Figure 9: Graph showing score calculated using various evaluation metrics for the best model on combined dataset

This graph shows score calculate by making use of various evaluation metrics namely, BLEU score, distance similarity and cosine similarity for the best model which in our case is our proposed model i.e LSTM using attention units. This was done for combined dataset i.e. we combined datasets accross various domains mentioned above into one complete dataset