Spring 2017

Headline Generation using Deep Neural Networks

Dhruven Vora
San Jose State University

Headline Generation using Deep Neural Networks

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Dhruven Vora

May 2017

The Designated Project Committee Approves the Project Titled

Headline Generation using Deep Neural Networks

by

Dhruven Vora

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2017

Dr. Jenny Lam	Department of Computer Science
Dr. Chris Pollett	Department of Computer Science
Dr. Thomas Austin	Department of Computer Science

ABSTRACT

Headline Generation using Deep Neural Networks

by Dhruven Vora

News headline generation is one of the important text summarization tasks. Human generated news headlines are generally intended to catch the eye rather than provide useful information. There have been many approaches to generate meaningful headlines by either using neural networks or using linguistic features. In this report, we are proposing a novel approach based on integrating Hedge Trimmer, which is a grammar based extractive summarization system with a deep neural network abstractive summarization system to generate meaningful headlines. We analyze the results against current recurrent neural network based headline generation system.

ACKNOWLEDGMENTS

I would like to show my gratitude towards my advisor Dr. Jenny Lam for guiding me in the research process for this Master's project. I would like to thank Dr. Chris Pollett for providing the DUC data set. I would like to Dr. Thomas Austin and Dr. Chris Pollett for being part of my graduate committee.

TABLE OF CONTENTS

CHAPTER

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction and Related Research

The advent of the internet has boosted the information consumption by users. Since the production of information has become easier, the rate of information flow has grown tremendously. This trend has made the selection of information difficult for the user. For example, there are thousands of news publications produce millions of news articles in a single day and to gain the attention of the reader some publishers generate catchy headlines with low content value. Since people generally make judgments solely reading the headline and skip the story, especially stories related to health, they may end-up doing wrong thing which may turn harmful. For example, an article published in Science Daily [1] under the title "*The benefits of chocolate during pregnancy*" was misleading. The article actually mentioned that "*This study indicates that chocolate could have a positive impact on placenta and fetal growth and development and that chocolate's effects are not solely and directly due to flavanol content.*" The article discusses the possibility of the positive effects of chocolate but headline has a different meaning.

An article headline is a short summary which captures the prime motive of the story. Radev [2] defines a summary as "A text that is produced from one or more texts, that conveys important information in the original text(s), and that is no longer than half of the original text(s) and usually significantly less than that [2]." Automatic text summarization is the task of generating the meaningful summary from the text. Our goal is to generate a headline which is more relevant to the story and should capture the meaning of the story being published.

There have been many approaches developed over the past half century to generate good quality summaries from the text. It started with the intention to generate good abstracts for the scientific research papers [2]. In the beginning, researchers focused on

linguistic features of the English text to find out which sentences are more important and included them in the summary. But it was hard for researchers to write a code that works with different types of writing styles of the authors. To make a generic summarization system algorithms were developed based on statistical properties of the language such as word frequencies [2]. But none of those attempts produced a good result.

The first attempt to use machine learning for summarization task was done by Kupiec et. al. in 1995 [3]. It was an extraction based approach where they calculate a probability of a sentence from the input text to be included in the summary. It was the first attempt to train the model using a dataset. They showed that machine learning based statistical approach works better than the standard grammar based approach. Later Zajic et.al. [4] used the Hidden Markov Models to calculate a morphological variation of the headline and the article and achieved a state of the art accuracy. But in the next year, Zajic [5] propose Hedge Trimmer, text summarization using the elements of the language such as semantics and grammar. Authors first generate a parse tree of the first sentence using BBN Parser. Then they generate extractive headlines by removing constituents from the parse tree until a threshold has been reached. Their approach suggests removal of the low information content of the sentence is done by using linguistically motivated heuristics. Authors showed that their algorithm beats the current state of the art HMM-based summarization algorithm mentioned above.

Until now the focus has been to extract content from the text as a summary. Extractive text summarization techniques rely on text features such as position in the text, word frequency, information content of the sentence to decide whether to add that portion of text in summary or not. Most commercially available summarization systems utilize the extraction based approach. They crop out important phrases from

the text and glue them together to generate the meaningful summary. Although the summary generates grammatical sentences, it fails to capture the meaning in fewer words.

Recently key advancements in the back-propagation algorithm have improved ways to train a neural network. Machine translation has become the favorite topic for researchers to test neural networks. Cho et. al. [6] used the encoder-decoder approach on two different models: recursive neural networks and recursive convolutional neural network. They learned that the former performs well on short sentences while the latter learns the grammatical structure better. Sutskever et. al. [7] implemented deep neural networks to learn the sequence to sequence mappings. The implementation used Long Short-Term Memory network, a newly developed variant of recursive neural networks. They achieved a very high quality of translation which matched the state of the art in automatic machine translation task.

Since text summarization can be viewed as a machine translation task where the input is verbose and the output is concise. This similarity between two different domains made it easy for researchers to apply the knowledge gained so far to solve summarization problem.

Konstantin Lopyrev [8] used a version of neural networks called recurrent neural networks to generate headlines from the article. The encoder transforms the input sentence into a distributed representation word at a time and fed to the hidden layers. The decoder takes the output of hidden layer and generates each word of a headline using attention mechanism. The algorithm uses attention mechanism to calculate importance of each word fed to the decoder. The author used the Global Vectors to find the most relevant word for the headline.

Rush et. al. [9] tried to implement a data-driven approach for summarization. Their approach incorporated less focus on the linguistic structure of the sentence i.e.

the don't make any assumption about the vocabulary of the sentence. The proposed model predicts the next word in the output sentence using a conditional language model on the input sentence and previously generated words in the output sentence. Before feeding to model, the input is encoded via Attention based encoder to generate a single representation of an entire input sentence. The language model used in the above-mentioned approach is based on [10].

Nallapati et. al. [11] used the recently developed sequence-to-sequence alignment which have been successfully used in many natural language processing tasks such as machine translation, speech recognition, video captioning. They applied the attention based encoder-decoder RNN which was originally developed for machine translation task. Authors restricted decoder vocabulary by only using words appearing in a given batch of documents. This large vocabulary trick has helped them to improve the quality of the result. Another trick is used to replace the unknown words in the summary with most relevant word from the source document.

Paraphrasing the text to condense the sentence works better and has a potential to generate summaries of quality humans can generate. This approach is called 'Abstractive text summarization'. Abstractive text summarizer needs to find the relation between words and the probability of occurrence of the word based on the context. Since the summary contains words which may not be present in the text, it cannot rely on the input text only. There needs to be a connection between occurrences of different words needs to be developed. Neural Networks are capable of building such connection through training.

Recent advancements in neural networks research such as recurrent neural networks, new ways to build language models, word vectorization and attention mechanisms make Neural Networks the most attractive choice for text summarization task. Since the domain of neural networks is not yet explored much, it has a lot of room for

improvement.

In this report, we explore a novel attempt to overcome the shortcoming of the neural networks based approach by using linguistic features of the text. We have used an extractive summarization method called Hedge Trimmer developed by David Zajic [5] to generate the temporary summary of the sentence and then feeding it to the recurrent neural network based system. We analyze the result against the baseline approach developed by Lopyrev Konstantin in [8].

CHAPTER 2

Background: Neural Network Approach to Text Summarization

The objective of the sentence summarization task is to generate a condensed summary from an input sentence. A sentence is a sequence of N words $\{x_1, x_2, ..., x_N\}$ and the summary generated is a sequence of M words $\{y_1, y_2, ..., y_M\}$ where M $<$ N. Neural networks are most suitable for this task since they build complex relationships between words with a simple interface.

Neural network expect the length of all inputs to be same as well as the length of all outputs. Hence in our case, the length of M and N is kept be fixed throughout.

2.1 Language Model

The goal of the neural network is to determine the probability distribution over a sequence of words. This probability distribution is called the Language Model. In a case of text summarization, the language model is used to predict the next word in the output given the probability of the incoming sequence of words. The probability of the t^{th} word y given the $t-1$ words generated and all the input words can be shown mathematically as follows,

$$\Pr(y_t|x_1, x_2, ..., x_N, y_1, y_2, ..., y_{M-1})$$

The probability of the summary is calculated as the sum of probabilities for all the output words given the occurrence of complete sequence of input words and previously generated output words,

$$\Pr(y_1, y_2, ..., y_M) = \sum_{t=1}^{T} \Pr(y_t|x_1, x_2, ..., x_{N-1}, y_1, y_2, ..., y_{t-1}) \quad (1)$$

Our goal is to maximize this probability. In other words, we want to minimize the error. The objective of the neural network is to minimize the following,

$$-\log \Pr(y_1, y_2, ..., y_M | x_1, x_2, ..., x_{N-1}) = -\sum_{t=1}^{T} \log \Pr(y_t | x_1, x_2, ..., x_{N-1}, y_1, y_2, ..., y_{t-1}) \tag{2}$$

2.2 One-hot vector

But actually, neural networks do not generate a word. They output probability of each word to be next in the sequence. Neural networks make use of an interesting concept of one-hot representation of a word. One-hot representation is a vector of V dimensions , where V is the size of the vocabulary of the input data. Each index represents a word and that index is fixed throughout the neural system. The input and output words are represented by one-hot vectors but there is a subtle difference between the internal representation. For input vector, only the index of the word has a value 1, rest all indices have a value 0. Output vector is a prediction vector, hence each index represents a fraction or probability of a word at that index. The sum of all probabilities is 1 in both the cases.

Neural Network-based language models are popular because of their ability to decrease the effect of the curse of dimensionality. Curse of dimensionality refers to the need for a huge number of training samples when learning complex functions [10]. When the number of input variables increases, the number of required examples can grow exponentially because huge number of different combinations are possible. e.g. if we want to generate a sequence of 10 words from the vocabulary of size 100,000 there are 10^{50} combinations possible. But to train a neural network with the same size of vocabulary, we need much lesser number of samples, around 10^{10}

2.3 Mathematics of Neural Network-based Language Model

So how does Recurrent Neural Network generate words or headline? The answer is, by chain rule of probability.

By chain rule of probability, we get the probability of next word w_{t+1} from previously generated words as follows,

$$\Pr(w_1, w_2, ..., w_t, w_{t+1}) = \Pr(w_{t+1}|w_1, w_2, ..., w_{t-1}, w_t)$$

But for very long sentences the probability multiplication turns out to be smaller enough such that it becomes difficult to calculate probabilities further. N-grams technique is used generally to restrict the context to a fixed size, where we consider the previously generated N words only instead of all generated words. We can calculate the probability for a word using a soft-max function shown by Bishop (1995) [12] as follows:

$$\Pr(w_t = k|w_{t-n+1}, ..., w_{t-1}) = \frac{e^{a_k}}{\sum_{l=1}^{n} e^{a_l}} \tag{3}$$

where,

$$a_k = b_k + \sum_{i=1}^{h} W_{ki} \tanh(c_i + \sum_{j=1}^{(n-1)d} V_{ij} x_j) \tag{4}$$

Here b, c and W, V are parameters (or weights) of the network. For each input sequence of words, neural network learns weight parameter V and adds some bias constant c. Then it applies tanh function on the internal state to find out which weights are crossing a threshold value. Those weights are again weighted with W and bias constant b. to generate the internal state of each word. The weights are updated using a gradient descent algorithm to maximize the log-likelihood.

$$\mathrm{L}(\theta) = \sum_{t} \log \Pr(w_t|w_1, w_2, ..., w_{t-1}) \tag{5}$$

Equation (2) shows the sum of log-likelihood over all words.

The gradient $\frac{\partial L(\theta)}{\partial \theta}$, shows the direction in which the model is being trained and is a hill-climb. The goal is to find the global minimum value of the θ such that the

change $\frac{\partial L(\theta)}{\partial \theta}$ tends to 0. This change can be computed by back-propagation algorithm by calculating a derivative of the equation (5) as proposed by Bengio [10]. There are other variants of the above equations proposed by [13] [14].

Since we calculate the probability of occurrence of a word based on the previous words and the input, this model can only generate a word which occurred with the other words in the input sentences of the training data. This model may work for extractive summaries but our goal is to generate abstractive summaries where the word in the summary may not have occurred with words in the input sentence. This is a big challenge. Neural nets rely on the algorithm that extracts the features of words and since these features are continuous-valued, they make optimization problem much simpler [10]. The representation of each word as a vector of features extracted is called 'distributed representation' [15] or 'word embedding'.

2.4 Word Embeddings

Word embedding is a feature learning technique in natural language processing where words are represented as a vector. This representation is helpful to find out the similarity between two words [16]. Two widely used word embeddings are GloVe and Word2Vec. GloVe is an unsupervised algorithm that generates a vector representation of words. The algorithm is trained on the aggregated global word-word co-occurence statistics from a corpus [17]. Word2Vec is a predictive model which uses skip-gram model for generating word embeddings [18]. Word2Vec is developed by Tomas Mikolov [18] while GloVe algorithm is developed at Stanford [17].

The advantage of GloVe over Word2Vec is that it is easier to parallelize the implementation. We chose GloVe to generate word embeddings for this reason.

2.5 Recurrent Neural Networks (RNN)

So the next question is, how to implement the language model that we just represented mathematically?

The answer is to use Recurrent Neural Networks. An RNN is a kind of Neural network which operates on sequential input. The task performed by each layer is same hence it is called ‘‘Recurrent’’. RNNs have a capability to store and carry the state through time. This ability to memorize the previous information makes RNN suitable for long inputs such as sentences or patterns.

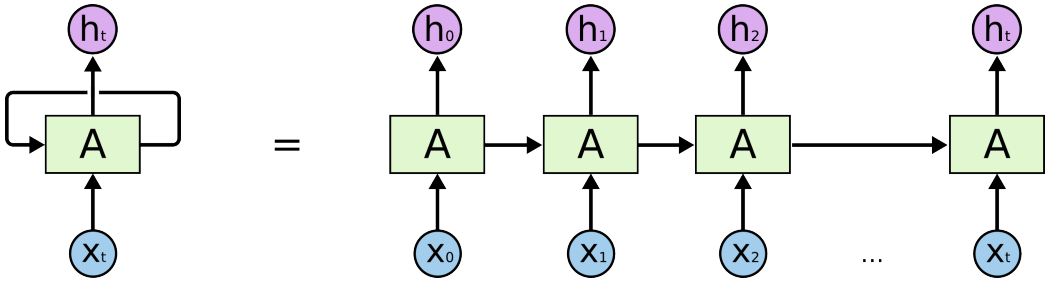


Figure 1: Unfolded Recurrent Neural Network

Source: *http://www.wildml.com*

Training RNNs is similar to training a traditional neural network except the back propagation algorithm, which is little different. Because the parameters are shared by different time-steps, the gradient at each output depends on the previous time-steps. e.g. To calculate the value of a gradient at the 4th time step, we need to back propagate 3 steps and sum up the gradients. This is called back-propagation through time (BPTT).

As the image shows, Box A is the implementation of equation (3) and (4) and the same box is updated each time we input a new word. In other way, one can say that the network is carrying the context from previous words through time. All x are input words while all h are generated words.

2.5.1 Vanishing Gradients

Though RNNs trained with BPTT are quite effective in learning the word dependencies in the sentence, they have difficulties in learning long range dependencies. In gradient based learning, as the number of time steps (number of words in a sentence) increases, the value of the gradient in the earlier time steps becomes smaller. Any change in the network output causes a very small change in the network parameters preventing the network from learning effectively. This problem is called vanishing gradients.

Vanishing gradients problem depends on the choice of the activation function. Some activation functions (even popular ones such as tanh or sigmoid) squash their input to a small output range usually in the range of [0,1]. This problem grows when we stack up multiple networks (time steps) over each other.

In this report, we are proposing an approach to solve this problem. There have been few developments in this area that have solved this problem up to certain extent. LSTM is the most successful and used variant of RNN used to reduce the vanishing gradients issue.

2.6 Long Short-Term Memory Networks (LSTM)

Long Short-Term Memory (LSTM) network, a flavor of RNN, can learn long-term dependencies. It was invented by Hochreiter and Schmidhuber [19]. LSTM maintain cell state, which is regulated by gates. Gates are the structures that optionally let information through them. Each box A from Fig. (1) calculates some more hidden context to pass the only relevant information to the next word.

LSTM contains 3 gates. First ‘‘forget gate’’ decides what information should not pass through and to be forgotten. Second layer ‘‘input gate’’ will update the values of the previous state. Third ‘‘add gate’’ adds new candidate values to the state.

The three gates are defined mathematically as follows,

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \text{ forget gate}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \text{ input gate}$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \text{ add gate}$$

Here, W_f, W_i and W_C are the weights that decide which information to forget, to remember and which word's probability should be updated. Colah [20] provides a detailed explanation of the internal working of LSTM.

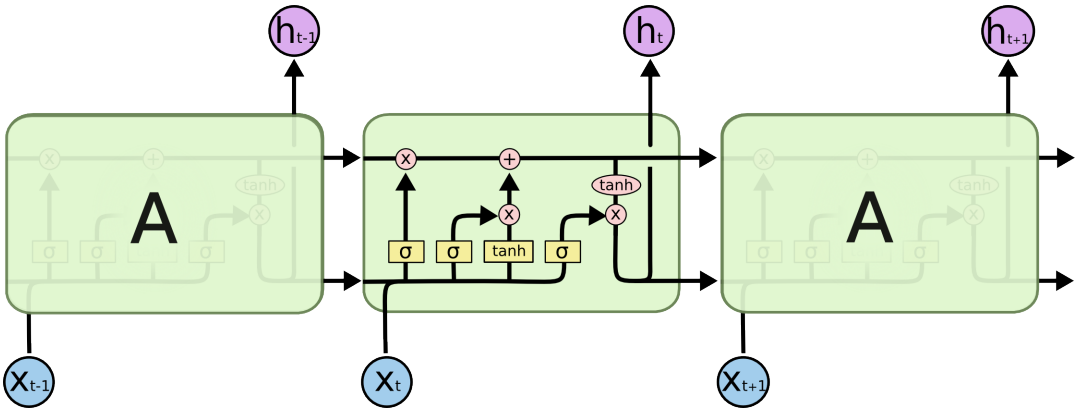


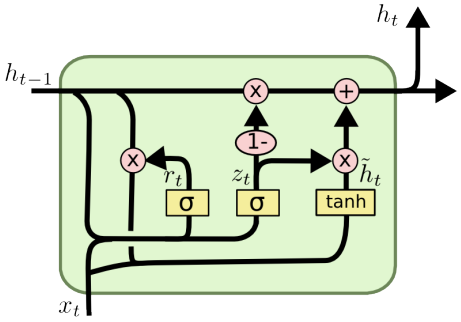
Figure 2: Long Short-Term Memory

Source: *http://www.wildml.com*

2.7 Gated Recurrent Units (GRU)

GRU (Gated Recurrent Units) is a newer but simplified version of an LSTM unit with fewer parameters. It can also remember long range dependencies and help to prevent vanishing gradient problem. The performance of GRU is found to be on par with LSTM [21].

GRU combines ‘‘forget’’ and ‘‘input’’ gates into a single ‘‘update’’ gate. It also merges cell state and hidden state. GRU consists of two gates. A reset gate that determine which part of old memory to keep and a update gate that updates the state at current time step.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 3: Gated Recurrent Units

Source: *http://www.wildml.com*

Colah [20] provides a detailed explanation of the internal working of GRU.

2.8 Attention Mechanism

Though we process the context information using the complex mechanisms as seen in LSTM and GRU, they are not enough to generate the high-quality output. In the last layer of the neural network, we perform attention mechanism to find the most appropriate word to select and which word to give attention to.

Attention mechanism helps the network remember certain aspects of the input better. The attention mechanism in neural networks is loosely related to the attention mechanism in humans. Humans generally focus on a certain region of the image as ‘‘High resolution’’ while the unimportant region is ‘‘Low resolution’’ [22].

Attention mechanism solves one of the important limitations of the recurrent neural networks. Recurrent neural networks expect an input sentence of the fixed length (here length means a number of words in a sentence). Encoding the information for a very long sentence is difficult. With attention mechanism, we no longer need to encode the full sentence and the model can focus on the features based on the output generated so far.

In his paper [8] Lopyrev Konstantin calculate the weight over each input word

and determine how much attention needs to be paid to each word. To compute the attention for t^{th} word when outputting $t' - th$ word is:

$$a_{y_{t'}}(t) = \frac{\exp(h_{x_t}^T h_{y_{t'}})}{\sum_t^T \exp(h_{x_t}^T h_{y_{t'}})})$$

Where h_{x_t} represents the last hidden layer generated after processing t^{th} input word.

Following is an example visualization of how attention mechanism focuses on the words:

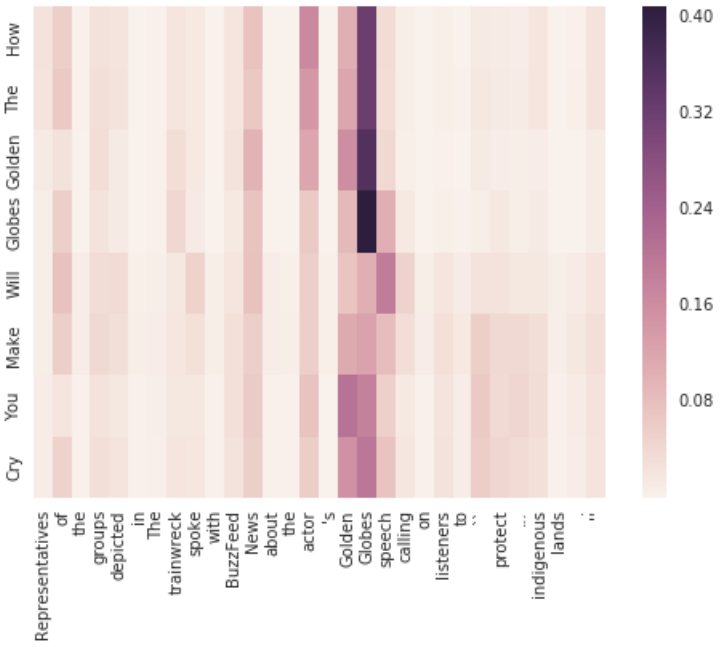


Figure 4: Heatmap of attention mechanism

Source: *https://github.com/udibr/headlines*

Here the x-axis shows the input sequence of words and the y-axis shows the output sequence (summary). Each shade shows attention weights between the input and output word. The darker cells show more attention to those input words is paid by the model to generate respective output word.

CHAPTER 3

Background: Hedge Trimmer approach to text summarization

Although the neural network based text summarizers produce good results, there have been grammar based summarizers for a long time. Following is one such algorithm, which produces results of good quality compare to other summarization systems.

Hedge Trimmer [5] is a headline generation system that uses linguistically motivated heuristics to select a potential headline. It generates the parse tree from a sentence and removes constituents from the parse tree until a length threshold has been reached. The algorithm relies on the grammatical structure of English language and identifies the components which even if removed does not alter the meaning of the resulting sentence.

The approach is described in brief as follows,

1. The sentence is parsed by the BBN parser [23]. The result is the parse tree with the phrases and words tagged (named entity tagging). For example, an independent phrase is marked with S node, while noun phrase is marked with NP and verb phrase is marked VP.

1. Choose left most S with NP,VP.

According to the author's observation, human generated headlines contains most of the information in the beginning of the sentence.

For example,

Input: Rebels agree to talks with government officials said Tuesday.

Parse: *[S* **[S [NP Rebels] [VP agree to talks with government]]** *officials said Tuesday.]*

Result: Rebels agree to talks with government.

2. Remove low content units

The algorithm removes some low-level determiners such as *a* and *the* and time expressions such as *yesterday* or *Friday* etc. The author observed that these low level

determiners do not contribute to the semantics of the sentence and even if they are removed from the headline, it will reflect the meaning of the news article with high quality.

For example,

Input: The State Department on Friday lifted the ban it had imposed on foreign fliers.

Parse: *[Det The]* **State Department** *[PP [IN on] [NP [NNP Friday]]]* **lifted** *[Det the]* **ban it had imposed on foreign fliers.**

Result: State Department lifted ban it has imposed on foreign fliers.

3. Iterative Shortening

Linguistically peripheral material such as extra phrases is removed from right side of the sentence. Another type of iterative shortening removes proposed adjuncts. It is done with the motivation that all of the human- generated headlines does not contain preamble of the story. For example, a noun phrase containing another noun phrase (NP over NP) in that case they remove the outer noun phrase from the sentence.

For example,

Input: A fire killed a firefighter who was fatally injured as he searched the house.

Parse: **[S** *[Det A]* **fire killed** *[Det a]* **[NP [NP firefighter]** *[SBAR who was fatally injured as he searched the house]* **]]**

Result: fire killed firefighter

In the above example, SBAR stands for Subordinate Clause that provides an extra information about the noun.

Algorithm 1 Hedge Trimmer

```
1: procedure TRIM
2:     fileptr ← read the pickle file containing articles and headlines
3:     loop: line ← fileptr.readNextLine()
4:         if len(line) > 1 then:
5:             ts ← create trimmer sentence
6:             ts ← applyRules(ts)
7:             writeOutput(ts)
```

Algorithm 2 Hedge Trimmer

```
1: procedure APPLYRULE(TS)
2:     trimmerSentence ← ts
3:     if useMultiRootS == true then:
4:         cand ← extract leftmost S node from trimmerSentence
5:         trimmerSentence.addCandidate(cand)
6:     if useMultiPreposedAdjuncts == true then:
7:         cand ← remove preposed adjuncts from trimmerSentence
8:         trimmerSentence.addCandidate(cand)
9:     if useMultiConjunction == true then:
10:        cand ← remove the similar conjunctions from trimmerSentence
11:        trimmerSentence.addCandidate(cand)
12:    if useSingleDeterminer == true then:
13:        cand ← keep single determiner from trimmerSentence
14:        trimmerSentence.addCandidate(cand)
15:    return trimmerSentence
```

We were provided with the jar (executable java) file by David Zajic, the author of the algorithm. It generates multiple candidate solutions. Selecting a potential input from the candidate sentences was difficult since many of them were closely related to each other. We converted the machine code to source code by reverse engineering and updated code to select a sentence randomly from the candidates.

CHAPTER 4

Overview of the proposed approach

In chapter (3) we discussed how text summarization is performed using Neural Network and chapter (4) describes the grammar based approach for headline generation. The grammar based approach does not make use of the context very well. Neural networks don't consider grammar or the sentence structure into consideration.

We propose a new approach which integrates both the approaches discussed. Our motivation to combine these approaches lies in the vanishing gradients problem faced by recurrent neural networks. According to our hypothesis, ''The Hedge Trimmer can shorten the sentence and bring the semantically important words closer to each other. By feeding the resulting sentence to neural networks, we will be able to mitigate the vanishing gradient problem and generate quality summaries.''

4.1 Overview

The approach is described in brief as follows,

1. We extract the first line of each article and feed it to Hedge Trimmer.

2. The output of the Hedge Trimmer is a shortened sentence which we feed to a recurrent neural network.

3. Headline generation using beam-search algorithm

We implemented the neural networks using the code developed by Ehud Ben-Reuven [24]. The author is using Keras library to implement LSTM. We used the code without any changes to generate our baseline result. We implemented GRU and Vanilla RNN using the code as a reference. We changed parameters of the neural networks such as initial values of weight vectors, a number of neural network layers and error rate calculation algorithm to suit our requirement.

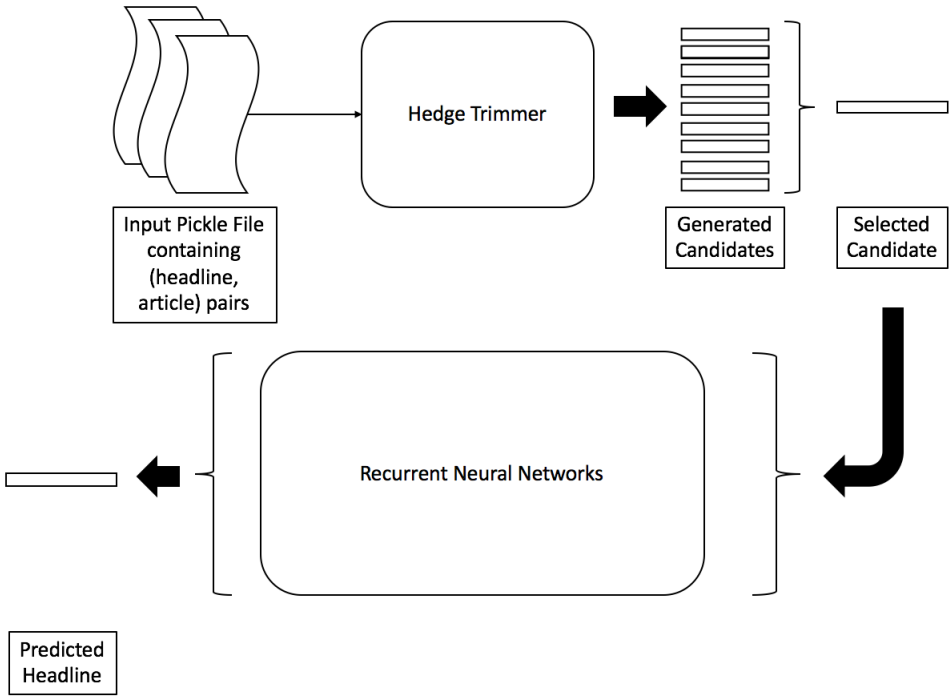


Figure 5: Proposed Texts Summarization System

We generated 3 different models of the neural network,

1. We built the first model using LSTM to understand the impact of the Hedge Trimmer on our baseline model.

2. The second model we built with GRU since GRUs are relatively new and weren't tested by the baseline model.

3. The third model we implemented using vanilla RNN to compare the performance of Hedge Trimmer without the support from LSTM layer.

Following is the pseudo-code of the algorithm.

Algorithm 3 Proposed Algorithm

procedure TRAIN
    Load embeddings and vocabulary mappings from vocabulary-embeddings file.
    $X,\ Y \leftarrow$ Load data (articles and headlines) from pickle file.
    $X \leftarrow$ *hedgeTrimmer.trim(articles)*
    $X_{train}, X_{test}, Y_{train}, Y_{test} \leftarrow$ Split data into train and test dataset
    model $\leftarrow$ Build a Neural network model (add customer attention layer)
    traingen $\leftarrow GENERATOR(X_{train}, Y_{train})$
    loop 500 times:
    model.fitGenerator(traingen)
    Dump model weights in a pickle file
    gensample()
procedure PREDICT
    article $\leftarrow$ read sentence
    summary $\leftarrow$ *model.predict(trimmedLine)*
    return summary

4.2 Stages of Training

In the following sections, we explain each stage in brief.

4.2.1 Data Collection and Cleaning

Our dataset comprises of articles with headlines. Before using the data we cleaned the data set. We removed Unicode characters and all non-alphabetical symbols since they are not adding any semantic information to the sentence and also they make it difficult for the neural network to understand the relationship between words. Then we make a pair of each headline with respective article's first sentence and dump it in the file. This file is fed to Hedge Trimmer to generate intermediate summaries (trimmed sentences).

4.2.2 Generating GloVe Vectors

Every word in the data set was converted into a vector of 50 dimensions using the GloVe library. This helped us identify and replace any words in the input sentence which were not present in the dictionary with the word present in the dictionary. Then we moved the generated vector file to Keras user directory.

4.2.3 Trimming

Once the data file was ready, it was fed into Hedge Trimmer. Hedge Trimmer is written in Java, hence we used JVM to run it. Trimming is only performed on the article and not on a headline. Once the trimming is complete, the pair of a headline and trimmed article is stored in a file.

4.2.4 Performing Training on Neural Networks

Once the trimming was complete we fed the generated (headline, article) pairs to train neural networks based model.

4.2.5 Evaluation

Since text summarization is considered as a type of translation from a verbose language to a concise language. It is hard to check the quality of the output since the quality of translation is subjective to the human judgment. For automatic evaluation the metric of measurement must be assigned quality scores to correlate with human judgment, it is considered as a benchmark and all the scores are tried to accurately depict the human judgment as close as possible. Benerjee et. al. [25] highlights that any good metric must possess following five attributes,

1. It must **correlate** highly with human judgment.
2. It must be **consistent**, giving similar results for similar text.
3. It must be **sensitive** to differences between texts.
4. It must be **reliable** that translation systems that score similar should perform similarly.
5. It must be **general** that it should work with wide range of text domains and scenarios.

There are multiple systems developed for scoring such as BLEU, NIST, METEOR, LEPOR etc. Wolk et. al. [26] provides a good comparison between these methods.

We are using BLEU [27] as an evaluation metric. It is the first metric to report high correlation with human judgment. It calculates the precision by comparing the similarity between the sentences based on the words that appear on the result as well as reference sentence.

CHAPTER 5

Data Set

To train the model we need data in the form of (headline, article) pairs. Since we are generating headline content from the first line of the paragraph of the article we extract the first sentence and pair it with the headline. Then we store these pairs in an external file. We also need to generate word embeddings and for that, We created a document by concatenating all the articles and headlines. This document is then fed to generate word embeddings.

We have used two different data sets: The first one is from Google DeepMind [28], which is an Artificial Intelligence company. They have open sources CNN News dataset. It contains around 92,000 articles with highlights or human-written summaries. A sample of an article is shown in appendix A. For the experiment, we have chosen the first paragraph of an article as my input because it has been observed that most relevant information for the headline is present in the first paragraph of an article [8].

For the second dataset, we retrieved around 600,000 news articles from Reuters news archive [29]. The archive contains news articles from the US and covers a wide variety of news. To get the data we wrote a crawler to download web pages from Reuters news website and extract the headlines and articles.

CHAPTER 6

Experiment and Evaluation

6.1 Experimental Setup

The experiment was performed on following system configuration:

1. AWS instance configuration: 1 Tesla K80 GPU, 4 CPU cores, 61 GB RAM.

2. Primary coding language: Python 2.7

3. Neural networks libraries: Keras 2.1 (Neural Networks Library), Theano 0.9.0 (Deep NN framework)

4. Other required libraries: Numpy, Scipy, python-Levenshtein libraries.

5. To run Hedge Trimmer Jar: Java 1.7

6.2 Reproducing baseline results

The first experiment we performed was to reproduce the results from a paper [8] with the CNN and Reuters data sets. The paper uses Gigawords [30], which was unavailable to us due to financial constraints. To compare the performance with the reference algorithm we performed 500 training iterations (same as the reference paper) on the complete dataset.

6.3 Training on LSTM

The second experiment we performed using a proposed model (with Hedge Trimmer) with LSTM as a neural network layer.

6.4 Training on GRU

The third experiment we performed using a proposed model (with Hedge Trimmer) with GRU as a neural network layer. This was an opportunity to test GRUs which are claimed to be on par with LSTMs.

6.5 Training on a Vanilla RNN

This experiment was performed to test the effectiveness of the proposed model with vanilla RNN (with Hedge Trimmer).

6.6 Testing on a generated Model

Once the training is completed, we store the model in the external file. The model can be reused this way without being generated again. We tested on data-set of size 10% of the training data.

CHAPTER 7

Results

7.1 Scores

Table 1 represents the BLEU scores obtained from experiments.

	CNN	Reuters
LSTM - Baseline	0.2248	0.2766
Hedge Trimmer + LSTM	0.2237	0.2539
Hedge Trimmer + GRU	0.2234	0.2495
Hedge Trimmer + Vanilla RNN	0.1830	0.2138

Table 1: BLEU scores

Since we tried to reproduce results on datasets different from the dataset mentioned in the reference paper, our baseline BLEU score may vary. Hence we compared all the subsequent experiments with respect to the score we obtained.

All the above models were trained for 500 iterations over same dataset.

7.2 Loss during training

The following results compares the error value changes during training of baseline model and the models trained on input trimmed by Hedge Trimmer.

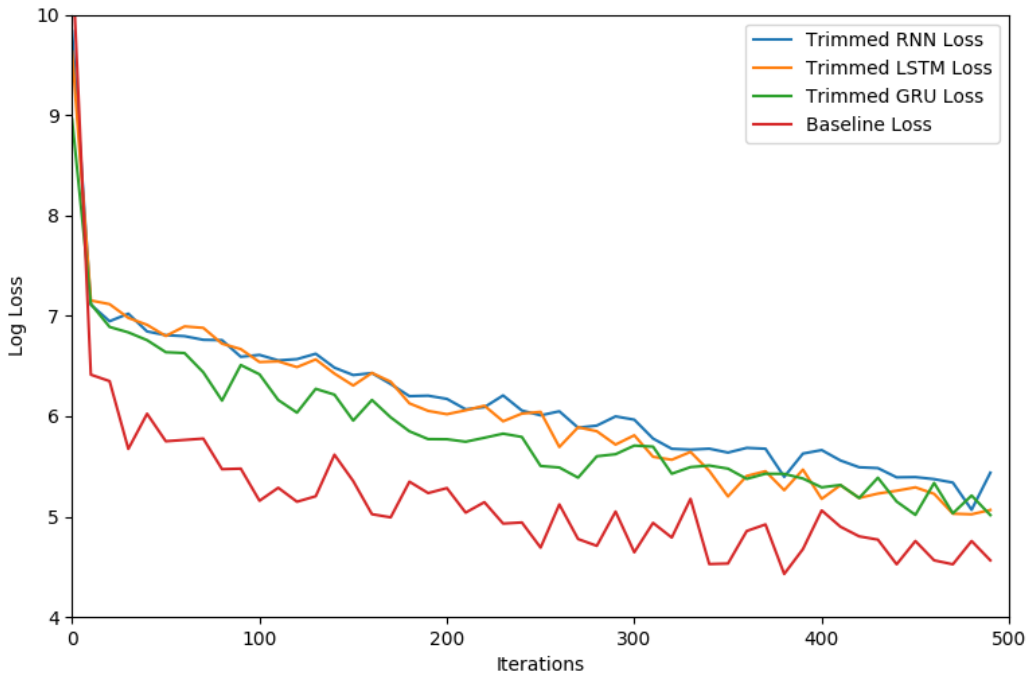


Figure 6: LSTM vs Trimmed LSTM training loss

7.3 Generated Samples

Following are the samples generated by Hedge Trimmer.

Input sentence	The output of Hedge Trimmer
two u.s. soldiers have been killed in iraq the u.s. military said on sunday pushing may to the brink of becoming the deadliest month for u.s. forces this year	two u.s. soldiers have been killed in iraq pushing may to brink of becoming deadliest month for u.s. forces year
tokyo jan 7 japanese stocks were down but off earlier lows on monday slumping on worries over the u.s. economy before bargain hunting emerged with defensives such as drugmaker eisai co ltd	tokyo jan 7 japanese stocks were down but off earlier lows slumping on worries over u.s. economy
feb 11 ferro corp forecast a fourth quarter profit below analysts ' expectations due to a manufacturing interruption and increased raw material costs across the company 's businesses.the chemicals and coating company	feb 11 ferro corp forecast a fourth quarter profit below analysts ' expectations due to a manufacturing interruption
new york july 20 benefiting from bad times management consultant accenture ltd is worth about 20 percent more than its shares are trading for barron 's said on sunday.an expert in technology	new york July 20 benefiting from bad times management consultant accenture ltd is worth about 20 percent more than its shares are trading for barron's
sees q3 gaap shr loss $ 0.15 to $ 0.20 sees q3 oper shr $ 0.70 to $ $ 0.75 says realized investment loss of $ 143 mln in q3 oct 20	sees q3 gaap shr loss $ 0.15 to $ 0.20 sees q3 oper shr $ 0.70 to $ 0.75 says realized investment loss of $ 143 mln in q3 oct 20

Table 2: Sample output of Hedge Trimmer

Following are the samples of testing data and the headline generated out of it.

Description	Actual headline	Generated headline
the most emotionless society is singapore 's despite its reputation for being among the world 's richest , a new survey has revealed	Wealthy Singapore ranks as world's most stoic nation	philippines , meanwhile , registers as most emotional nation
as she approaches the fifth anniversary of the accident that paralyzed her , record-breaking paralympic gold medallist mallory weggemann is in defiant mood .	Paralympian Mallory Weggemann rebuilds her life piste by piste	her next challenge is scuba diving before climbing mount kilimanjaro with her father
(cnn) president barack obama will renew his push for paid sick days and paid family leave for the millions of american workers who do n't have either during a visit to baltimore on thursday , according to valerie	Obama to press for paid sick days and paid family leave	the white house chose linkedin to unveil the president 's plans

Table 3: Test samples for CNN News data

Description	Actual headline	Generated headline
two u.s. soldiers have been killed in iraq the u.s. military said on sunday pushing may to the brink of becoming the deadliest month for u.s. forces this year.one was killed by	factbox military and civilian deaths in iraq	factbox **:** military and civilian deaths in iraq
tokyo jan 7 japanese stocks were down but off earlier lows on monday slumping on worries over the u.s. economy before bargain hunting emerged with defensives such as drugmaker eisai co ltd	japan stocks **off lows bargain hunters emerge**	japan stocks **edge down on u.s. economy fears**
feb 11 ferro corp forecast a fourth quarter profit below analysts ' expectations due to a manufacturing interruption and increased raw material costs across the company 's businesses.the chemicals and coating company	**ferro** sees q4 profit below **wall street** view	**update** sees q4 profit below **analysts '** view
new york july 20 benefiting from bad times management consultant accenture ltd is worth about 20 percent more than its shares are trading for barron 's said on sunday.an expert in technology	**accenture little known and undervalued** barron 's	**former ceo says growth barron** barron 's
sees q3 gaap shr loss $ 0.15 to $ 0.20 sees q3 oper shr $ 0.70 to $ $ 0.75 says realized investment loss of $ 143 mln in q3 oct 20	update 1 **insurer wr berkley sees q3 hurt by investment losses**	**rpt update** update 1 **pentagon q3 to q3 q3 loss**

Table 4: Test samples for Reuter data

7.4 Analysis

The behavior of neural networks is hard to understand because it shows different behavior for different input sentences and it is hard to quantify this relation. Due to time constraint, we could not do a deeper analysis of the results but we made few observations from the above results.

1. In some cases, neural networks add words in the headline which are not relevant to the actual story. We observed that Neural Network picks words which appear frequently in the corpus with some of the words present in the actual story.

2. More number of samples increases the overall accuracy of the summarizer. We observed from the experiments that headlines generated from CNN articles are not meaningful as well as relevant to the actual story. But this is not the case for Reuters data set. This helps us understand one of the limitations of neural networks. Neural networks generate a stronger correlation between words that appear in the same context frequently.

Ideally, in the real world, this may not be the case. Some words with high information value may occur less frequently. For example, words such as 'Accenture', 'Ferro' appears only a few times in the dataset. This makes it difficult for neural networks to build an association of other words with these words.

3. We hypothesized that Hedge Trimmer may help us generating high-quality headlines but it seems that picking up only important words did not make any significant difference in the performance of the neural networks. It seems that by trimming the input sentence, we actually narrowed down the context for Neural Network to generate appropriate sentence. It can be seen that the model trained using Hedge Trimmer with either LSTM or GRU, beats the Vanilla RNN implementation but still falls short of baseline performance.

CHAPTER 8

Conclusion and Future Work

In this report we presented a new attempt to perform text summarization using linguistic features along with a neural network. Although the solution has shown less potential than the current state of the art, it gives a useful insight into the behavior of the neural network.

There exists an opportunity to improve the model by exploring more ways to integrate grammar information in the neural networks to gain the benefits of both domains. Also, there is a huge opportunity to improve the quality of the summary and extend the summary to paragraph level from sentence level.

LIST OF REFERENCES

[1] ''The benefits of chocolate during pregnancy,'' February 2016. [Online]. Available: https://www.sciencedaily.com/releases/2016/02/160201214629.htm

[2] R. D. R., E. Hovy, and K. McKeown, ''Introduction to the special issue on summarization.'' *Computational linguistics*, 2002.

[3] K. Julian, J. Pedersen, and F. Chen., ''A trainable document summarizer.'' in *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, 1995.

[4] D. Zajic, B. Dorr, and R. Schwartz, ''Automatic headline generation for newspaper stories,'' in *In The Proceedings Of The ACL Workshop On Automatic Summarization/Document Understanding Conference (DUC)*, 2002.

[5] B. Dorr, D. Zajic, and R. Schwartz, ''Hedge trimmer: A parse-and-trim approach to headline generation,'' in *Proceedings of the HLT-NAACL 03 on Text Summarization Workshop - Volume 5*, 2003.

[6] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, ''On the properties of neural machine translation: Encoder-decoder approaches,'' *CoRR*, 2014.

[7] I. Sutskever, O. Vinyals, and Q. V. Le, ''Sequence to sequence learning with neural networks,'' *Advances in neural information processing systems*, 2014.

[8] L. Konstantin, ''Generating news headlines with recurrent neural networks,'' *arXiv preprint*, 2015.

[9] A. M. Rush, S. Chopra, and J. Weston, ''A neural attention model for abstractive sentence summarization,'' *CoRR*, 2015.

[10] Y. B. et. al., ''A neural probabilistic language model,'' *Journal of Machine Learning Research*, 2003.

[11] R. Nallapati, B. Xiang, and B. Zhou, ''Sequence-to-sequence rnns for text summarization,'' *CoRR*, 2016.

[12] B. C. M., *Neural networks for pattern recognition*. Oxford university press, 1995.

[13] S. H. and G. J.-L., ''Training neural network language models on very large corpora,'' in *Joint Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2004.

[14] M. F. and B. Y., ''Hierarchical probabilistic neural network language model,'' *AISTATS*, 2005.

[15] H. G.E., ''Learning distributed representations of concepts,'' in *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, 1986.

[16] J. Pennington, R. Socher, and C. D. Manning, ''Glove: Global vectors for word representation,'' in *EMNLP*, 2014.

[17] J. Pennington, R. Socher, and C. D. Manning, ''Glove: Global vectors for word representation,'' August 2014. [Online]. Available: https://nlp.stanford.edu/projects/glove/

[18] e. a. Mikolov Tomas, ''Distributed representations of words and phrases and their compositionality,'' *Advances in neural information processing systems*, 2013.

[19] H. Sepp and J. Schmidhuber, ''Long short-term memory,'' *Neural computation*, vol. 9.8, pp. 1735--1780, 1997.

[20] C. Olah, ''Understanding lstm networks,'' August 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[21] J. C. et. al., ''Empirical evaluation of gated recurrent neural networks on sequence modeling,'' *CoRR*, 2014.

[22] D. Britz, ''Attention and memory in deep learning and nlp,'' January 2016. [Online]. Available: http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/

[23] e. a. Miller, Scott, ''A novel use of statistical parsing to extract information from text,'' in *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*. Association for Computational Linguistics, 2000.

[24] E. Ben-Reuven, ''Headlines,'' March 2017. [Online]. Available: https://github.com/udibr/headlines

[25] S. Banerjee and A. Lavie, ''Meteor: An automatic metric for mt evaluation with improved correlation with human judgments,'' in *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, 2005.

[26] K. Wołk and D. Koržinek, ''Comparison and adaptation of automatic evaluation metrics for quality assessment of re-speaking,'' *arXiv preprint arXiv*, 2016.

[27] K. e. a. Papineni, ‘‘Bleu: a method for automatic evaluation of machine translation,’’ in *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002.

[28] H. et al., ‘‘Deepmind q and a dataset,’’ 2015. [Online]. Available: http://cs.nyu.edu/~kcho/DMQA/

[29] ‘‘Reuters site archives,’’ 2017. [Online]. Available: http://www.reuters.com/resources/archive/us/

[30] C. Napoles, M. Gormley, and B. V. Durme, ‘‘Annotated english gigaword,’’ November 2012. [Online]. Available: https://catalog.ldc.upenn.edu/ldc2012t21

APPENDIX

Dataset sample

A.1 CNN News

Following is the example of an article from CNN news,

(CNN) -- It was a much-hyped meeting between two of golf's biggest names -- and the chaotically enthusiastic crowd in China were not let down by Rory McIlroy and Tiger Woods in a surreal showdown worth a reported $2 million.

Billed as the "Duel at Jinsha Lake," Monday's clash coincided with the launch of a multi-million-dollar housing project which is being built around the course in Zhengzhou, the capital of Henan province.

Fans turned out in their thousands to witness the world's top two golfers -- one of them, Woods, the man largely responsible for the game's increased worldwide popularity, and the other, the 23-year-old McIlroy, its big hope for the future.

@highlight

World No. 1 Rory McIlroy beats Tiger Woods in exhibition match in China

@highlight

"Duel at Jinsha Lake" takes place in front of chaotic scenes in Zhengzhou

@highlight

Fans throng the course, models decorate the tees while a luxury yacht cruises the lake

A.2 Reuters news

Following is the example of an article from Reuters news,

UPDATE 1-TSMC plans five new advanced wafer plants -paper

TAIPEI Jan 2 TSMC (2330.TW) plans to build five new advanced 12-inch wafer plants on the island in the next few years, a local newspaper said on Tuesday, after a government move to allow companies to make more advanced chips in China.

On the plants' completion, Taiwan Semiconductor Manufacturing Co. Ltd. (TSMC) (TSM.N), the world's top contract chip maker, would have a total of seven 12-inch factories in Taiwan, the Economic Daily News quoted unidentified officials at the economics ministry as saying.