Sentence Compression for Text Summarization (Natural Language Processing)

Lum Yao Jun
lumyaojun@gmail.com

Abstract

In this paper, we attempt sentence compression by using extractive text summarization. By formulating the task as a multi-label deletion-based problem, we are thus able to train a deep learning model to solve this task. With fewer training examples, we achieve similar F1-Scores to past papers and can generate summaries which are mostly grammatically coherent and usable. A POC of the model was used on a proprietary dataset owned by the author's workplace with similar results.

1. Introduction

In this era where the amount of textual data is growing at an exponential rate, it is increasingly important to develop ways to efficiently process and derive insights from such data. As it is tedious and difficult for humans to extract key points from large volumes of text, automatic text summarization is necessary for reducing the overall time spent on processing textual data. Text summarization aims to shorten long pieces of text, creating a coherent and fluent summary highlighting only the main points of the text. Traditionally there are two approaches to the problem:

1) Extractive text summarization involves pulling key words or phrases from the source text and combining them to make a summary

2) Abstractive text summarization involves paraphrasing and shortening parts of the source document to generate a summary

In this paper, extractive, instead of abstractive, text summarization is used, as it is difficult to programmatically evaluate an abstractive summary due to the lack of a constraint on the type of vocabulary and phrases able to be used. Previous methods to conduct extractive text summarization include creating a summary with words with the highest TFIDF scores, and the TextRank algorithm, which extracts sentences most representative of the text based on similarity scores between sentences.

We instead take a different approach, training a sentence compression based textual summarizer using deep learning techniques. This thus takes the form of a multi-label deletion-based task, where the model picks which words to keep from the original text, while aiming to preserve the grammatical coherence and important content of the original text. Summaries generated with our method are thus naturally a subsequence of the original text.

The project outlined in this paper was initiated with the intention of applying a POC to be used in the author's workplace for processing textual documents.

2. Related works

In the sentence compression literature, many approaches rely on the presence of additional synthetic information for sentences as inputs. This is done to minimize chances of introducing grammatical mistakes in the output. [4]. For example, Filippova & Altun [1] rely on the pruning of dependency trees, and Zhao & Aizawa [2] use part-of-speech tags and word dependency relations as features in a reinforcement learning framework. The most recent work in this area by Kamigaito & Okumura [3] uses additional language model features alongside parent-child word relationships for sentence compression, achieving the current state of the art F1 score of 0.855 on the Google sentence compression dataset. Such methods, however, require said synthetic information for model training, which requires either be manual tagging by humans or generation with a separate model. Additionally, systems depending on such synthetic information are vulnerable to error propagation should there be errors in the generation of said information.

As an alternative, Filippova et al. [4] proposed a compression model using only tokens, without access to other linguistic information, using LSTMs to output summaries. Apart from the data available in the Google sentence compression dataset, Filippova et al generate an additional 2 million sentence compression pairs and use 3 layers of unidirectional LSTMs to achieve an F1 score of 0.80. This approach by Filippova et al does not require additional synthetic information like POS tags in the training data. In this paper, we attempt to use a similar method to Filippova et al. [4], using a purely token based approach to achieve sentence compression on the Google sentence compression dataset introduced by Filippova & Altun [1].

3. Dataset and Features

We use the Google dataset produced by Filippova & Altun [1] for pretraining the model. This dataset contains 210,000 examples of sentence compression pairs based on online news articles. Data is publicly available at https://github.com/google-research-datasets/sentence-compression/tree/master/data.

Original Text:

"Serge Ibaka -- the Oklahoma City Thunder forward who was born in the Congo but played in Spain -- has been granted Spanish citizenship and will play for the country in EuroBasket this summer, the event where spots in the 2012 Olympics will be decided."

Summary:

"Serge Ibaka has been granted Spanish citizenship and will play in EuroBasket"

Figure 1: Example of sentence compression pair present in Google dataset

Figure 1 displays an example sentence compression pair from the Google dataset. All word summaries constructed in this dataset consist of words present in the original sentence, arranged in the word order of the original sentence. This allows for the model training task to generate purely extractive summaries, where compression is a subsequence from the original sentence.

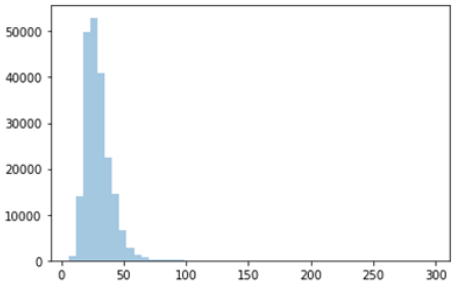

Figure 2: Histogram of original sentence lengths

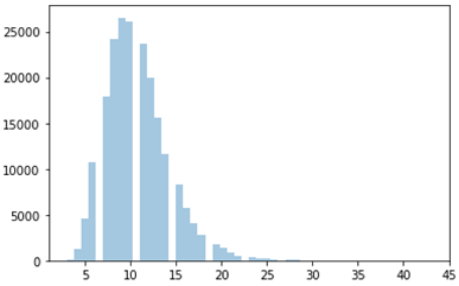

Figure 3: Histogram of summary sentence lengths

Figures 2 and 3 describe the frequency of the original sentences and associated summaries, respectively. 210,000 original sentences with an average word length of 29.7 are paired to associated summaries with average word length of 10.7. The average compression rate (Number of words reduced divided by original sentence length) of this dataset is 60.47%.

While the dataset also includes additional features such as POS tags of individual words, we do not use any additional information apart from the actual text and associated summary.

3.1. Encoding of input text

To encode individual words of sentences into a machine-readable format for training in our baseline models, we convert each word in the original sentence into 300-dimension GloVe embeddings [6]. The pretrained GloVe embeddings map words into a vector space with meaningful linear substructures, providing better word representations based on word co-occurrences. As the available training data (210,000 examples) in the Google dataset is small in comparison to the dataset used by Filippova et al (2 million examples), we use the GloVe embeddings to supplement additional semantic information for words in the inputs to our baseline models. For similar reasons, we use the embedding layer of a pre-trained language model for our final model.

3.2. Preparation of labels

To prepare labels for our models, we generate a one-hot encoded [Equation 1] vector the length of each original sentence $\boldsymbol{N}$ for each sentence compression pair.

$$1_A(w_n) := \begin{cases} 1 & \text{if } w_n \in A, \\ 0 & \text{if } w_n \notin A. \end{cases} - [1]$$

Each element of the one-hot vector is a binary indicator for the nth word of a sentence w_n, with true values signifying the presence of the word in a sentences' summary ($\boldsymbol{A}$).

To ensure comparability of model results with previous works, we use the first 10,000 sentence compression pairs as our testing set, and the remaining 200,000 pairs for training our models.

4. Methods

Figure 4 displays the general model architecture used for all models in this paper. The general model architecture comprises 3 parts: (i) the embedding layer, (ii) the model hidden layers, and (iii) the sigmoid output layer.

Individual words from each original sentence are tokenized and fed into the embedding layer at either word or wordpiece levels. The embedding layer then creates vector representations of these inputs, which are then fed into the model hidden layers, which differ depending on the

model used. Finally, a sigmoid layer generates an output score between 0 and 1, used to determine if an individual word should be kept in the sentence summary. Each word of the sentence will have a corresponding output from the

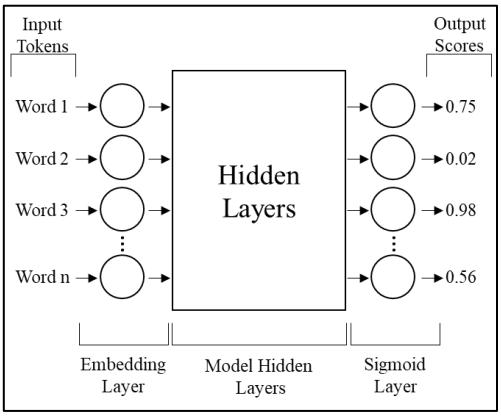

Figure 4: General Model Architecture

sigmoid layer. The sentence compression problem is thus formulated as a multi-label classification task, where each individual input token is classified into being kept or not.

4.1. The Baseline Model

Our Baseline Model uses GloVe embeddings in the Embedding Layer to encode our input text. We then add a single LSTM layer, a max pooling layer and a feed forward layer to make up the Model Hidden Layers. The structure of the LSTM layer helps to model both long- and short-term dependencies amongst words in a sentence. It does this by holding an internal "cell state" which is updated as words of a sentence are fed into the layer. Each "cell" of the LSTM layer consumes the output "hidden state" and "cell state" of each previous cell alongside each new word, and determines a new "hidden state" and "cell state" using input, forget, and output gates.

$$\begin{aligned} i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\ f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\ g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\ o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \quad -[3]$$

Equation 3 dictates the formula for the LSTM layer, where t is each subsequent word in a sentence (representing the "time step"), $\boldsymbol{h_t}$ is the layer's hidden state at t, $\boldsymbol{c_t}$ is the cell state at t, $\boldsymbol{h_{t-1}}$ is the layer's hidden state at t-1 or an initialized hidden state at time 0, $\boldsymbol{i_t}$ is the input gate, which determines how to update the cell state based on the new information, $\boldsymbol{f_t}$ is the forget gate, which determines what information to remove from the previous hidden state and current input, and $\boldsymbol{o_t}$ is the output gate, which together with the cell state determines the next hidden state. σ is the sigmoid function, and $\odot$ is the element-wise product of two matrices.

4.2. The Bi-LSTM Model

The second model we use is like the baseline model in using GloVe embeddings in the Embedding Layer. However, we now use a 2-layer bi-directional LSTM to feed into a max pooling and feed forward layer for our Model Hidden Layers. Bi-directional LSTMs work similarly to the regular LSTM used in the Baseline Model. However, while a regular LSTM only parses a sentences' words from beginning to end, bi-directional LSTMs also parse words from end to beginning. This is advantageous as the internal cell states of the bi-directional LSTM can now preserve information from both before and after a certain sentences' word, allowing for a better representation of sentence context.

4.3. The Pre-Trained BERT Model

While we first train a model from scratch in the Baseline and Bi-LSTM models, we instead tap upon pre-trained language models in our final model. Specifically, we use the pre-trained BERT language representation model for fine tuning and transfer learning on our dataset [5]. The BERT model is pre-trained on 2 tasks: (i) Masked Language Modeling (predicting the value of randomly masked words in the input sentence, and (ii) Next Sentence Prediction (given a pair of 2 sentences, determine if the second sentence follows after the first in the original text document) [5]. The two tasks, trained over the BooksCorpus (800M words) and English Wikipedia (2500M words), allow the final model to contain rich semantic information on the English language. On its release, the BERT model obtained state-of-the-art results in 11 natural language processing tasks [5].

In our final model, we no longer use GloVe embeddings in our Embedding layer, and instead use the pre-trained Embedding layer from BERT. This differs from previous models as BERT's embedding layer uses WordPiece embeddings instead of words and encodes each wordpiece into a 768-dimensional vector [5].

In our Model Hidden Layers, our BERT implementation stacks 12 separate groups of layers, with each group containing a self-attention layer feeding into 3 feed forward layers, alongside normalization and dropout layers after the 1st and 3rd feed forward layers. Each self-attention layer generates a 768-dimensional representation for each wordpiece in a sentence based on specific wordpiece representations in the previous layer. This essentially creates a weighted representation of each wordpiece in a sentence based on the various wordpiece representations of the previous layer. All model parameters for the Embedding

and Model Hidden Layers are pre-loaded with weights from the base-uncased version of the BERT model.

4.4. Loss Function and optimizer

All models are trained to minimize the Binary Cross Entropy loss function [Equation 2] across all output scores in a sentence, for all sentences in the training data.

$$\ell(x, y) = \text{mean}(L),$$
$$L = \{l_1, \ldots, l_N\}^\top, \quad -[2]$$
$$l_n = y_n \cdot \log x_n + (1 - y_n) \cdot \log(1 - x_n)$$

The implementation of the Binary Cross Entropy function in Equation 2 compares the sigmoid outputs x_n with the corresponding binary labels y_n and is lower the closer $\boldsymbol{x_n}$ is to 1 when y_n is 1.

The Adam optimizer [7] takes into account momentum and the RMSProp algorithm to accelerate learning speed and is used for all models in this paper.

4.5. Padding of sentences

To standardize the inputs of the sentences in our models, we pad our original sentences to a length of 150 words per sentence. Original sentences with over 150 words are truncated at the end. Wordpiece lengths are padded or truncated to a length of 200 respectively.

5. Results and discussion

Various hyperparameter choices were explored in the training of the models. A grid search was first used to test learning rates from 0.1 to 1 x 10^{-6} . Eventually, implementing an optimizer learning rate scheduler which decreases the learning rate from 3 x 10^{-5} , according the cosine function, with 3 hard restarts was used due to superior results on the validation set.

The maximum batch size allowed by available GPU memory was used, for quicker model training time and faster iterations. This resulted in a batch size of 128 for Baseline and Bi-LSTM models, and 16 for the BERT pre-trained model.

The number of epochs each model was trained was determined by visually inspecting a plot of the training and validation loss. Optimal epochs were chosen at where validation loss started to plateau or diverge from the decrease in training loss. This is to avoid overfitting to the training set. Other methods used to avoid overfitting were addition of dropout to the LSTM and BERT models.

Table 1 displays the results of the 3 models alongside the score of randomly initialized labels for comparison. It can be seen that even the baseline model provides a significant performance boost in comparison to randomly choosing words to keep in the summary. However, the baseline model is only able to achieve an F1 score of 0.585.

The Bi-LSTM model presents a performance gain over the Baseline model with a F1 score of 0.604. However, these low F1 scores result in poor-quality predicted summaries. Visual analysis of the predicted summaries show that not only do both the Baseline and Bi-LSTM model not produce grammatically coherent sentences, they also are prone to several types of errors (See Appendix A for a selection of examples from the test set displaying various types of errors).

The first error prevalent in these models are that predicted summaries do not end in grammatically coherent ways. Table 2 shows an example of this, where the Bi-LSTM prediction ends with an adjective without an associated noun.

Model	Layers	Epochs	Precision	Recall	F1-Score
Random	Random initialization of labels, mean of 100 sample examples.	NA	0.037	0.500	0.069
Baseline	• Embedding layer using GloVe • Unidirectional LSTM layer with hidden size 300 • Max-pooling layer • Feed Forward Layer RELU activation, size 300 • Feed Forward Layer Sigmoid activation, size 200	20	0.707	0.500	0.585
Bi-LSTM	• Embedding layer using GloVe • 2 bidirectional LSTM layer with hidden size 300 • Max-pooling layer • Feed Forward Layer RELU activation, size 300 • Feed Forward Layer Sigmoid activation, size 200	20	0.728	0.516	0.604
BERT	• BERT Wordpiece embedding layer • 12 BERT layer groups, each group having a self-attention layer, 3 Feed Forward Layers and a Normalization layer • Feed Forward Layer Sigmoid Activation, size 200	10	0.838	0.805	0.821

Table 1: Model Results

Predictions generated by the Baseline and Bi-LSTM models also tend to do poorly on sentences with summaries not consisting of earlier words in the sentence. Table 3 shows an example of a sentence compression pair with this characteristic, where earlier words in the sentence are not relevant to the summary.

Original	Five people have been taken to hospital with minor injuries following a crash on the A17 near Sleaford this morning.
Summary	Five people have been taken to hospital with minor injuries following a crash on the A17 near Sleaford.
Bi-LSTM Prediction	five people have been taken to hospital with minor
BERT Prediction	five people have been taken to hospital following a crash .

Table 2: Example where Bi-LSTM is unable to end summary well

Original	At a ceremony held in Charlotte, NC, Monday night, Pitt defensive tackle Aaron Donald won the 2013 Bronko Nagurski Trophy.
Summary	Aaron Donald won the 2013 Bronko Nagurski Trophy.
Bi-LSTM Prediction	at trophy .
BERT Prediction	aaron donald won the 2013 bronko nagurski trophy .

Table 3: Example where earlier words are not used in summary

Original	This 118-carat egg-sized jewel, the largest diamond ever sold at auction, fetched $30.6 million at Sotheby's in Hong Kong yesterday, or $259,322 per carat.
Summary	This jewel, the largest diamond ever sold at auction, fetched $ 30.6 million.
Bi-LSTM Prediction	jewel , the largest diamond ever sold at auction
BERT Prediction	jewel fetched $ 30 . 6 million in hong kong .

Table 4: Example where numbers are needed in summary

Finally, the Baseline and Bi-LSTM models are unable to report numbers properly. Table 4 demonstrates an example where numbers are dropped from the summary entirely.

Various architectural choices were explored to improve model performance. However, both increasing the number of layers and increasing the size of each layer resulted in the training loss plateauing beyond the first epoch. While previous authors have proven that having more data would improve performance [4], it was also difficult to obtain more data for training, given the lack of publicly available alternatives

The BERT pre-trained language model was thus used to combat these issues with the earlier models and lack of training data. BERT was chosen in order to harness the semantic information gained in it's pretraining to our sentence compression task. As seen in the Tables 2-4, the model based on Fine-Tuning BERT is able to produce more grammatically coherent sentences while avoiding the previously identified pitfalls.

	Predicted Negative	Predicted Positive
Actual Negative	158243	17180
Actual Positive	21621	89073

Figure 5: Confusion matrix for BERT model

Figure 5 displays the word-level Confusion Matrix showing the actual labels vs predicted outputs of the BERT Model. It can be seen from Tables 2-4 that while the BERT model is unable to fully predict the original summary, its predictions are mostly grammatically coherent and usable. The F1-Score for the BERT model is 0.821, similar to the results of Filippova et al. [4]. However, Filippova et al. [4] use an additional 2 million sentence compression pairs, while we are able to achieve the same results with a much smaller dataset of 200,000 examples.

6. Conclusion

In this paper, we attempt sentence compression by using extractive text summarization. By formulating the task as a multi-label deletion-based problem, we are thus able to train a deep learning model to solve this problem. While much of the previous literature and current state of the art models rely on having additional synthetic information as inputs, we instead fine-tune a model with text as the only training data, without additional linguistic information.

While models trained from scratch did not perform well on the dataset, using the pre-trained BERT language model for fine tuning produced the best results. With fewer training examples, we achieve similar F1-Scores to past papers like Filippova et al. [4] and can generate sentences which are mostly grammatically coherent and usable as summaries.

With further work, results can still be improved. Filippova et al. [4] have shown that having a bigger set of training data would better improve model performance. Additionally, recent advancements in the Natural language Processing Community have resulted in more advanced models than BERT with improved results.

Finally, while not covered in this paper due to confidentiality purposes, the model was deployed to an additional 592 paragraphs of a proprietary dataset containing economic reports owned by the author's workplace, with good results (0.782 F1 Score, attributable to differences in the domain of the datasets). Going forward, we will be implementing this model in the workplace to distil various articles for easy consumption.

7. Results and discussion

All steps involved in this project were conducted by Lum Yao Jun only.

8. References

[1] Filippova, K., & Altun, Y. (2013). Overcoming the Lack of Parallel Data in Sentence Compression. *EMNLP*.

[2] Zhao, Y., Luo, Z., & Aizawa, A. (2018). A Language Model based Evaluator for Sentence Compression. *ACL*.

[3] Kamigaito, H., & Okumura, M. (2020). Syntactically Look-Ahead Attention Network for Sentence Compression. ArXiv, abs/2002.01145.

[4] Filippova, K., Alfonseca, E., Colmenares, C.A., Kaiser, L., & Vinyals, O. (2015). Sentence Compression by Deletion with LSTMs. *EMNLP*.

[5] Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv, abs/1810.04805*.

[6] Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

[7] Kingma, D. P., & Adam, B. J. (2017). A method for stochastic optimization. cornell university library. arXiv preprint arXiv:1412.6980.

[8] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. the Journal of machine Learning research, 12, 2825-2830.

[9] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., … Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d extquotesingle Alch'e-Buc, E. Fox, & R. Garnett (Eds.), Advances in Neural Information Processing Systems 32 (pp. 8024–8035). Curran Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[10] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Brew, J. (2019). Transformers: State-of-the-art Natural Language Processing. arXiv preprint arXiv:1910.03771.

[11] Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras

[11] Oliphant, T. E. (2006). A guide to NumPy (Vol. 1). Trelgol Publishing USA.

[12] McKinney, W., & others. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51–56).

[13] Loper, E., & Bird, S. (2002). NLTK: the natural language toolkit. arXiv preprint cs/0205028.

Appendix A: Sample Predictions

Sample Predictions for Baseline Model

#	Original Sentence	Summary	Predicted Summary
A1	Five people have been taken to hospital with minor injuries following a crash on the A17 near Sleaford this morning.	Five people have been taken to hospital with minor injuries following a crash on the A17 near Sleaford.	five people have been taken to hospital with
A2	Several school districts in Hampton Roads are holding classes this Presidents' Day to make up for days missed because of the snow.	Several school districts are holding classes this Presidents ' Day to make up for days missed.	several school districts in hampton roads are holding classes
A3	Markets continued to remain under pressure on Thursday morning as financial heavyweights like ICICI Bank, HDFC, and HDFC Bank declined by 1-2% each.	Markets continued to remain under pressure.	markets continued to remain under pressure on
A4	At a ceremony held in Charlotte, NC, Monday night, Pitt defensive tackle Aaron Donald won the 2013 Bronko Nagurski Trophy.	Aaron Donald won the 2013 Bronko Nagurski Trophy.	at a
A5	This 118-carat egg-sized jewel, the largest diamond ever sold at auction, fetched $30.6 million at Sotheby's in Hong Kong yesterday, or $259,322 per carat.	This jewel, the largest diamond ever sold at auction, fetched $ 30.6 million.	this 118 jewel , the largest diamond ever

Sample Predictions for Bi-LSTM Model

#	Original Sentence	Summary	Predicted Summary
B1	Five people have been taken to hospital with minor injuries following a crash on the A17 near Sleaford this morning.	Five people have been taken to hospital with minor injuries following a crash on the A17 near Sleaford.	five people have been taken to hospital with minor
B2	Several school districts in Hampton Roads are holding classes this Presidents' Day to make up for days missed because of the snow.	Several school districts are holding classes this Presidents ' Day to make up for days missed.	several school districts in hampton roads are holding classes this
B3	Markets continued to remain under pressure on Thursday morning as financial heavyweights like ICICI Bank, HDFC, and HDFC Bank declined by 1-2% each.	Markets continued to remain under pressure.	markets continued to remain under pressure on
B4	At a ceremony held in Charlotte, NC, Monday night, Pitt defensive tackle Aaron Donald won the 2013 Bronko Nagurski Trophy.	Aaron Donald won the 2013 Bronko Nagurski Trophy.	at trophy .
B5	This 118-carat egg-sized jewel, the largest diamond ever sold at auction, fetched $30.6 million at Sotheby's in Hong Kong yesterday, or $259,322 per carat.	This jewel, the largest diamond ever sold at auction, fetched $ 30.6 million.	jewel , the largest diamond ever sold at auction

Sample Predictions for Pre-Trained BERT Model

#	Original Sentence	Summary	Predicted Summary
C1	Five people have been taken to hospital with minor injuries following a crash on the A17 near Sleaford this morning.	Five people have been taken to hospital with minor injuries following a crash on the A17 near Sleaford.	five people have been taken to hospital following a crash .
C2	Several school districts in Hampton Roads are holding classes this Presidents' Day to make up for days missed because of the snow.	Several school districts are holding classes this Presidents ' Day to make up for days missed.	school districts are holding classes this presidents ' day .
C3	Markets continued to remain under pressure on Thursday morning as financial heavyweights like ICICI Bank, HDFC, and HDFC Bank declined by 1-2% each.	Markets continued to remain under pressure.	markets continued to remain under pressure .
C4	At a ceremony held in Charlotte, NC, Monday night, Pitt defensive tackle Aaron Donald won the 2013 Bronko Nagurski Trophy.	Aaron Donald won the 2013 Bronko Nagurski Trophy.	aaron donald won the 2013 bronko nagurski trophy .
C5	This 118-carat egg-sized jewel, the largest diamond ever sold at auction, fetched $30.6 million at Sotheby's in Hong Kong yesterday, or $259,322 per carat.	This jewel, the largest diamond ever sold at auction, fetched $ 30.6 million.	jewel fetched $ 30 . 6 million in hong kong .