EAI Endorsed Transactions
on Internet of Things
Research Article EAI.EU

Sentence Fusion using Deep Learning

Sohini Roy Chowdhury[1], Kamal Sarkar[2,*]

[1,2] Department Of Computer Science & Engineering, Jadavpur University, Kolkata ,India

Abstract

The human process of document summarization involves summarizing a document by sentence fusion. Sentence fusion combines two or more sentences to create an abstract sentence. Sentence fusion is useful to convert an extractive summary to an abstractive summary. The extractive summary contains a set of salient sentences selected from a single document or multiple related documents. Redundancy creates problems while creating an extractive summary because it contains sentences whose segments or phrases are redundant. Sentence fusion helps to remove redundancy by fusing sentences into a single abstract sentence. This moves an extractive summary to an abstractive summary. In this paper, we present an approach that uses a deep learning model for sentence fusion. which is trained over a large dataset. We have tested our approach through both manual evaluation and system evaluation. The result of our proposed approach shows that our model is good enough to fuse sentences effectively.

Keywords: Abstractive Summarization; Deep Learning; Sentence Fusion

Received on 06 October 2023, accepted on 10 December 2023, published on 14 December 2023

doi: 10.4108/eetiot.4605

1. Introduction

Text summarization aims to produce a condensed version of a long document with salient information. In present times, just with a click on the world wide web, we are flooded with information. Hence having a crux from vast information is really very challenging task. Text summarization provides a solution to this information overflow problem. Humans summarize a long document by electing important concepts from the document. Human applies some cognition procedure for reformulating salient concepts of the document to produce an abstract. Since human cognition is not computable, an exact imitation of the human ways of document abstraction is very difficult to be achieved by a machine. The abstractive summarization process generates human-like a summary that may contain new words which are not present in the input. When human creates a summary, they focus on salient sentences which are often fused to create a concise summary.

The earliest approach to abstract generation [36] on text summarization used local and global trimming of the extracted sentences for producing an abstractive summary, (Barzilay and McKeown, 2005) [3] proposed a parse and fuse approach for sentence fusion to produce an abstractive summary. The target of this work was to create a grammatically correct non-redundant sentence by fusing multiple sentences. Many other researchers have used compression rules for deleting sentence constituents to compress sentences. They have used either anaphoric constraints [4] or integer linear programming [6] to maintain cross-sentence coherence. The text generation-based approach [1][5][2] is another kind of approach that selects textual units from the input and used a surface realization step to generate new sentences.

In the neural network-based approaches [7][8], authors have used sentence extraction and sentence compression by rewriting using a complex neural network architecture. Recently, the transformer-based approach has shown unprecedented success in the domain of text generation. A transformer-based model used encoder-decoder architecture [34][35] where the encoder accepts input sentences, and the decoder generates an abstract based on the input sentences. One of the problems with the transformer-based model is that

*Corresponding author. Email: jukamal2001@yahoo.com

it has an input length restriction that truncates the input to some limit. This leads to difficulty in processing a longer text document as a whole. To combat this situation, sentence selection and sentence fusion can be useful.

In this method, the extracted sentences can be carefully paired and fused to obtain an abstract sentence. Thus, sentence fusion is useful for summarizing long documents using the transformer-based model. In this paper, we focus on sentence fusion using a text-to-text transformer model. Our main contribution is not to develop a new transformer model. Our contribution here is to create a sentence fusion dataset, train a text-to-text transformer model for sentence fusion and evaluate sentence fusing tasks using the automatic evaluation method and human evaluation method.

Our paper is arranged in the following way. Related works are discussed in section2. The methodology is explained in section 3. Section 4 highlights the evaluation metric and results. Section 5 finally concludes the paper.

2. Related works

To date, various methods for sentence fusion have been proposed. We have categorized the existing sentence fusion methods into two groups:

- Sentence fusion using syntactic representation and text generation
- Sentence fusion using deep learning

2.1 Sentence fusion using syntactic representation and text generation

Sentence fusion plays an important role in abstract generation. (Barzilay and McKeown, 2005) [3] proposes a technique to effectively fuse multiple sentences and the generated fused sentence contains salient information from multiple sentences. Common information is identified by developing an approach to align syntactic trees for the input with paraphrase information. Subtrees of input sentences are created, and subsets of generated subtrees are matched. It is done by a bottom-up approach which performs multisequence alignment. A fusion lattice is constructed by combining fragments and enclosing the alignment along with lattice linearization to a sentence with a language model. (Bing et al.,2015) [5] proposes a method to fuse sentences using the selection and merging of phrases. Integer linear optimization was used to select salient phrases from a pool of information presented by phrases in the input corpus. First noun phrases and verb phrases are extracted, and scores of phrases are computed for the identification of salient concepts. Next, from that score new sentence is generated by selecting important phrases from different source sentences and merging them. (Liao, Lebanoff and Liu, 2018) [1] used Abstract Meaning Representation (AMR) to generate new sentences as abstract. AMR presents sentence meaning using a rooted, directed acyclic graph. In this graph nodes are represented as concepts and edges are represented as semantic relations. Source sentence selection, content planning, and surface realization were performed to generate an abstract. The first step, source sentence selection elects alike sentences with different aspects of the topic from the corpus. The next summary graph is generated from similar sentences in the content planning phase. Finally, a summary graph is converted to an abstract in the surface realization step. (Chenal and Cheung, 2016) [2] performs aggregation tasks to create new output sentences. First sentence fragments like a meaningful part of the input sentences are clustered. Next new dataset is created with different granularities from pre-existing sentences. A user study is performed for confirmation of the validity of datasets and creation of two gold standard clusterings. These gold standards were used as an evaluation method. Finally, a clustering model using logistic regression and hierarchical clustering is created for performing this task. (Cheung and Penn,2014) [12] used unsupervised sentence enhancement for summarization. Firstly, sentence enhancement is performed by joining sub-trees of sentences to output sentences. Hence relevant information which is not alike to the input sentence can be added during fusion. Next, it is analyzed how the generation task incorporates text which is not present in the input but in the article with a similar topic. (Liu, Flanigan, Thomson, Sadeh, Smith, 2015) [15] used semantic representation to create an abstract. AMR graph is created from input text. Then a summary graph is created from that graph and finally abstract is generated from the summary graph. It is a structured prediction algorithm that transforms semantic graphs of the input into a single summary semantic graph. AMR graphs is created using a concept merging step, where coreferent nodes of the graphs are merged; next in the sentence conjunction step, the root of each sentence's AMR graph is connected to a dummy "ROOT" node. Next an optional graph expansion step, where additional edges are appended for creating a fully dense graph on the sentence level. These results in creating a single connected source graph. A Subset of the edges and nodes from the source graph is elected for incorporation in the summary graph. This is a condensed presentation worthiest semantic content from the input. (Mehdad, Carenini, Tompa, NG,2013) [14] created an entailment graph for selecting sentences. In the entailment graph, nodes are noted as linked sentences, edges are the entailment relation among nodes The entailment relation is described as if two sentences infer the same meaning it is called bidirectional entailment and one of the sentences is removed. If one sentence is more informative than the other, which is denoted as unidirectional entailment, the less informative sentence is removed. If both sentences contain some important information, then both sentences are included. Finally, sentence fusion was done with a word graph. (Nayeem, Fuad, Chali,2018) [16] proposed a sentence fusion-based abstract generation approach. In this approach, first, a sentence generation model is created which performs both the sentence fusion and paraphrasing by skip-gram word embedding model. Next, it is applied to the creation of a multi-document summarization system. This model is

unsupervised and has the ability to generate new worthy words as output.

2.2 Sentence fusion using deep learning

Recent approaches to sentence fusion tasks are mostly neural network-based and transformer-based approaches. (Chen and Bansal,2018) [7] proposed a model to perform rewriting of sentences. In this model, the first extractor agent elects salient sentences from the long corpus and then the abstractor network performs the rewriting of extracted sentences. Finally, policy-based reinforcement learning (RL) is used to combine two networks. (Mendes et al.,2019) [8] performed abstraction by using Long Short-Term Memory (LSTM). Summary-worthy sentences are first chosen from the input then LSTM architecture is used to create the summary state representations. Important information is provided by this state to increment summaries based on the previously extracted information. It also generates variable-length summaries. (Gehrmann, Deng, Rush,2018) [26] produced a bottom-up attention model. First, a selection mask is chosen from the input document then a neural model is constrained by this mask. The selection task is framed as a sequence-tagging problem, to detect summary-worthy tokens from input. Masking is employed to constrain copying words for performing bottom-up attention. (Lebanof, Song, Liu,2018) [28] performs an investigation of a novel adaptation method for the encoder-decoder framework. This approach was the first to couple the maximal marginal relevance (MMR) algorithm with pointer-generator networks to perform summarization. Importance and redundancy scores are computed for the source sentence. Sentences with the highest MMR scores are computed and finally, pointer generator decoder is used to identify important words to include in the abstract. (Celikyilmaz, Bosselut, He, Choi,2018) [22] proposed deep communicating agents to perform abstractive summarization. This approach used encoder-decoder architecture. Encoding is performed by collaborating with multiple encoder agents where each agent is assigned to a different section of the text. After encoding they broadcast it to others so that everyone can share global information in different sections of the document. This process is repeated at multiple layers. After encoding, the encoded information is passed to the decoder with a novel contextual agent attention. Finally, reinforcement learning is employed to generate concise abstracts. (Tan, Wan, Xiao, 2017) [25] produced graph-based attention neural model. This approach highlights generating worthy information from a document using the neural model. This is an encoder-decoder architecture along with an attention mechanism for generating abstract. Challenges regarding giving long sequences as input and producing long sequence output are investigated. A hierarchical decoding algorithm was proposed to produce an abstractive summary. (See, Liu and Manning,2017) [21] produced a pointer generator network to generate abstract. Firstly, a hybrid pointer-generator network is used to copy words from input by pointing. This helps to reproduce accurate information along with retaining the ability to generate new words by the generator. Next, a coverage mechanism is used to monitor which part is summarized already so that no repetition occurs. (Song, Zhao, Liu, 2018) [20] described transformer-based method for abstract generation incorporating seq2seq learning. In this approach dependency parse tree structure is combined with copy mechanism to include important words in summary. A structure-infused copy mechanism is used to copy source words along with their relations to the summary by their semantic and structural importance in the source sentences. Here GLOVE embedding was used to convert input word into vectors. (Raffel et al.,2020) [23] explored transfer learning techniques by introducing a framework that converts all text-based language problems into a text-to-text format. This work is a survey of transfer learning techniques. It discusses a comparative study among pre-training objectives, architectures, unlabelled data sets, transfer approaches, and language understanding tasks. (Lewis et al.,2020) [24] proposed BART, a denoising autoencoder for pretraining sequence-to-sequence models. In this model, pretraining is done in two steps, firstly, random noise function is added to text then seq2seq model is learned to regenerate the original text. Here a Transformer-based neural machine translation architecture is used. This architecture can also be renamed as bidirectional encoder, GPT (with the left-to-right decoder). This model is very effective for text-generation tasks. In our work, we have used a transformer-based approach to generate fused sentences as abstract. After passing input through the embedding layer, it is passed to six encoders and six decoders via a self-attention mechanism. Finally, the fused output is generated which is well understood and able to capture salient features from the input.

3. Methodology

3.1. Sentence fusion using text-to-text transformer model:

The architecture of a transformer model used for our sentence fusion task is shown in Figure 1.

A transformer model has two different stacks, an encoder stack, and a decoder stack. Though most of the transformer models follow encoder-decoder architecture [34], some exceptions exist there like the GPT family of models which can also be used for summarization by some modification. We have used mT5, a multilingual version of T5. T5 is a transformer model used for text-to-text generation.

A mapping is done by encoder, from a sequence $(x_1,x_2,\ldots\ldots x_n)$ to a continuous representation $z = (z_1,z_2,\ldots\ldots z_n)$[34]. Decoder can produce the output $(y_1,y_2,\ldots\ldots y_m)$from z with an element at a time. It is a regressive model. The basic building blocks of a transformer for text-to-text generation are discussed in this subsection. The basic building blocks are:

- Embedding Layer
- Positional Embedding Layer
- Encoder
- Decoder

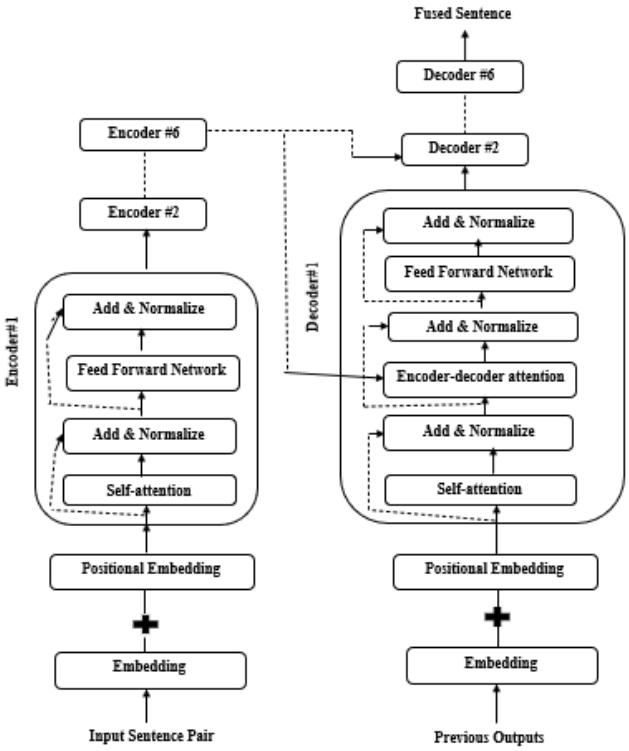

Figure 1. Architecture of transformer model

Embedding Layer

The first input sentence pair is passed to the embedding layer by converting it into a vector. Embedding layer maps input tokens to the sequence of vectors. The purpose of this layer is to capture semantic information of input words. Neural networks can understand numbers hence mapping of each input word to a vector with continuous values is done to represent that word.

Positional Embedding Layer

Positional information plays an important role to understand the input. It defines the semantics of input. In the case of Recurrent Neural Network (RNN) ordering of the input word is considered. Sequentially input is parsed word by word through this layer. But the encoder in the transformer has no recurrence like recurrent neural networks so it is needed to incorporate positional information via this layer. As it is not captured by the embedding layer, hence it is appended with the output of the embedding layer.

Encoder

The transformer model consists of six encoders. Each encoder is connected with each other. Each encoder has the following components.

Self-attention:

Output of positional embedding layer is passed to self-attention layer. This attention is known as intra attention as it considers the position of the input sequence to compute the representation of that sequence. Self-attention layer compares input sequence numbers with each other, and the corresponding position of the output sequence is modified. Both the output and input have the same dimension and sequence length.

Add & Normalize:

Output of self-attention layer and feed-forward network layer is passed to this layer. Add and normalize is two-step process. Add step is the residual connection. It is used to solve the vanishing gradient problem. The output of a layer is added with input in the add step. The Normalize step is for layer normalization. Residual connection is incorporated with two sub-layers after performing layer normalization. Layer normalization is defined as

LayerNorm(x+Sublayer(x)) Where, Sublayer(x) denotes function of sub-layer.

Feed Forward Network:

Output of add and normalize layer is passed to this layer. The purpose of this layer is to transform the attention vector in such a form that is acceptable to next encoder or decoder. Attention vectors are accepted once at a time in this layer. These attention vectors are independent of each other. Hence parallelization can be applied here. Output is processed by this layer from an attention layer to better-fit input for the next attention layer. During summary generation, humans pay attention to salient parts of the long document. Here attention function can be defined as the mapping of a query and a set of key-value pairs to an output, where the keys are the query, and the output are all vectors. Output is a weighted the sum of the values, where the weight is computed by a compatibility function among the query and the corresponding key. Multihead attention permits the model to share information among different sub-spaces. In the "encoder-decoder attention" layers, the previous decoder layer and memory layer produce queries, and the encoder generates the values. This allows the decoder to pay attention to all positions in the input.

Decoder:

There are six decoders in the transformer. The output of the last encoder is passed to the second decoder and encoder-decoder layer of the first decoder. Decoder also consists of a self-attention layer, add and normalize layer and feed-

forward network layer. Along with these a new layer is added here which is named as "encoder-decoder attention".

Encoder-Decoder Attention:

It generates multihead attention over the encoder output. Here also a residual connection is used around two sub-layers. An improved version of self-attention is used to prevent attention to subsequent positions by positions. An encoder-decoder Transformer is built up with the encoder, where input sequence is given, and the decoder, which produce the output sequence. A “fully visible” attention mask is used by the encoder. It allows a self-attention mechanism for paying attention to each input of its output. In the decoder, the self-attention mechanism uses a masking pattern such that the future is unseen to the model during the generation of output.

3.2. Model implementation

Each training example for the model is an input sentence pair and a fused sentence. Input sentence pair consists of two sentences separated by ## symbol. As shown in Figure 1 the pair of input sentences is submitted to the encoder via embedding followed by positional embedding layers. During training the corresponding fused sentences are considered as gold which is used to compute the loss. Thus, the model is trained on our training data consisting of 127897 training instances. The trained model is saved, and the saved model is tested on test data. We have used huggingface mT5 model[2].

3.3. Training dataset creation:

The transformer model used for sentence fusion needs a large amount of training data. The manual creation of a large amount of training data is a tedious task. We have used an automatic method for creating the dataset for our sentence fusion task. The method for data creation is described in this section. Each training instance consists of an input sentence pair and an output fused sentence. We have used Multinews summarization[3] for dataset creation. A Multinews dataset consists of 56216 document summary pairs. In this dataset, each summary is an abstract. We assume that a sentence in an abstract is linked to multiple sentences in the source document. Given a sentence in a training abstract, we find two nearest sentences from the source document. These two sentences are considered as the input sentence pair and the sentence taken from the abstract is considered as the corresponding fused sentence. Thus, a training instance is created automatically. The idea of automatic instance creation is given in Figure~3. For finding the semantically closest sentence of any given abstract sentence from the source document, we have used BERT and BERT (Bidirectional Encoder Representations from Transformers) embedding to obtain vectors of sentences and cosine similarity measures. The length of the vector obtained using BERT is 768. Cosine similarity is the cosine of the angle between two vectors, which means the dot product of two vectors divided by the product of their lengths. Cosine similarity is calculated using equation 1, where A and B are two sentence vectors. Figure~2 represents a sample training set.

$$\text{Cosine-similarity}=\frac{A.B}{||A||||B||} \quad (1)$$

Sample Training Input	Sample Training Output
In dissent, Justice Elena Kagan wrote of the big impact of the decision. "There is no sugarcoating today's opinion.”	In dissent, Justice Elena Kagan wrote "there is no sugarcoating today's opinion.”
Jeff Hallock said at a news conference. “We're embarrassed," Hallock said.	Jeff Hallock said “we’re embarrassed” at a news conference.
It had 845,000 YouTube views and counting by Saturday afternoon. You can watch the video on YouTube.	It's currently closing in on 8 million views in YouTube.
A hearing date has not been set. A date has not been scheduled.	No date has been set for the hearing.
President Donald Trump weighed in minutes after the decision was handed down, while Alito still was reading a summary of it from the bench. Impact of this decision was big.	President Trump weighed in minutes after the decision was handed down, while Justice Samuel Alito still was reading a summary of his majority opinion from the bench.

Figure 2. Sample training set

[2] https://huggingface.co/huggingface-course/mt5-small-finetuned-amazon-en-es

[3] https://huggingface.co/datasets/multi_news/viewer/default

Algorithm 1 Training data set creation

Input: Document summary pairs.

Output: A collection of instances where each instance consists of a sentence pair and the corresponding fused sentence.

1: **for** each document-summary pair **do**
2: Break the input document and the corresponding summary into sentences. Obtain sentence vector using BERT.
3: **for** each sentence S in an abstract **do**
4: Compute the cosine similarity of S with each sentence in the source document.
5: Rank the source sentences based on cosine similarity values from the highest to the lowest order.
6: Choose the highest similar two sentences from the source document as the sentence pair and consider the sentence S as the fused sentence.
7: **end for**
8: **end for**

Figure 3. Algorithm for training data set creation

4. Evaluation and Results

4.1. Evaluation metric:

We have evaluated our approach on the ROUGE metric and the BLEU metrics.

ROUGE:

ROUGE [32] measures n-gram overlap between a system-generated summary and the reference summaries [33]. In our case, we have used one reference abstract (one abstract sentence) for each system-generated fused sentence. ROUGE counts various kinds of overlapping units between the system summary and the reference summaries. We have used the latest version of the ROUGE package - ROUGE 1.5.5 for evaluating the system summaries. The ROUGE toolkit reports various ROUGE–N scores, for example, ROUGE-1, ROUGE-2, etc. Along with ROUGE-1 scores, many state-of-the-art summarization systems have been evaluated using ROUGE-2 (bigram-based), and ROUGE-SU4 (skip bigrams with skip distance up to 4 words [32]). We use ROUGE-F score scores to evaluate and compare our proposed neural summarization method with other existing summarization methods.

BLEU:

BLEU [30][31], represents Bi-Lingual Evaluation Understudy. It is the most popular metric and effective enough for measuring the performance of a Machine Translation (MT) system. It works on counting matched n-grams with reference text and system text. Bleu score is mathematically defined in equation 2.

$$\text{BLEU score}=\text{BP} * \exp\left(\sum_{n=1}^{4} \frac{1}{n} P_n\right) \quad (2)$$

BP represents Brevity Penalty. If the Machine Translation is short in comparison with the Reference translations, it penalizes the score. The mathematical expression for Brevity Penalty is given as follows:

$$\text{Brevity Penalty} = \min\left(1, \frac{\text{Machine Translation Output Length}}{\text{Maximum Reference Output Length}}\right) \quad (3)$$

P_n can be defined as follows:

$$P_n = \frac{\sum n\text{-grams count in Machine Translated Text}}{\sum n\text{-grams count in Reference Text}} \quad (4)$$

4.2 Results

Applying the automatic method on the Multinews dataset consisting of 56216 document summary pairs, we have created 127897 training examples for the sentence fusion model and 25579 test examples. Each train and test example has an input pair and a fused sentence. The proposed model is trained on the training dataset and its performance is tested on the test dataset is shown in Table~1.

Table 1. System evaluation on test data generated by automated process

ROUGE Score			BLEU Score			
R-1	R-2	R-L	1 gram	2 gram	3 gram	4 gram
0.4299	0.3298	0.4113	0.3250	0.2784	0.2655	0.2570

Human evaluation:

For model evaluation, we have also used a small test dataset created manually. This dataset consists of 50 input sentence pairs. Three judges were appointed for creating a

fused sentence from the input sentence pair. The result of system performance over human assessment on 50 documents is shown in terms of both the ROUGE metric and BLEU metric in Table~2.

Table 2. Evaluation of system performance using human provided abstract

ROUGE Score			BLEU Score			
R-1	R-2	R-L	1 gram	2 gram	3 gram	4 gram
0.4359	0.3493	0.4077	0.3405	0.2899	0.2738	0.2628

Keeping fused sentences created by Human 1 as a peer, fused sentences generated by Human 2 and Human 3 are considered a gold standard and compared with Human 1 using BLEU and ROUGE metrics. The same process is repeated for other human judges. This is done to evaluate the performance of each human with respect to others. The Results are shown in Table~3.

Table 3. Performance evaluation of human judges

ROUGE Score				BLEU Score		
Human	**R-1**	**R-2**	**R-L**	**1 gram**	**2 gram**	**3 gram**
Human 1	0.629	0.547	0.594	0.398	0.364	0.344
Human 2	0.732	0.673	0.700	0.536	0.519	0.482
Human 3	0.627	0.537	0.585	0.371	0.338	0.318

4.3 Model Parameter Tuning

The hyper parameters used during training the model are shown in Table 4.

Table 4. Hyper parameters used in model training

Parameter Name	Value
optimizer name	AdamWeightDecay
initial learning rate	5.6e-05
end learning rate	0.0
weight decay rate	0.01
training precision	mixed float16
Train Loss	3.1301
Validation Loss	2.6937
No. Of Epoch	4

4.4 Error Analysis

Some garbage output we have got from our model which is shown in Figure 5. On the other hand, some good output generated by our system is shown in Figure 4.

Test Input: When his mother died in 2011, he said in court docents that he had no income even though he was already supporting a wife and a son, according to the Times. Lome Leibel describes himself as a well-known Ferrari man and famous racer.

System Output: After his mother's death, he said he had no income at all.

Test Input: There have been no arrests. Birmingham police have not made any arrests in the shooting.

System Output: No arrests have been made by Birmingham police.

Figure 4. Sample Good Output Generated by System

Test Input: Tweet with a location You can add location information to your Tweets, such as your city or precise location, from the web and via third-party applications. You always have the option to delete your Tweet location history.

System Output: You can always choose to never delete your Tweet location history.

Test Input: The wounded officer is Crystal Almeida, 26. The third victim was identified as Scott Painter, a Home Depot loss-prevention officer.

System Output: The wounded officer is Crystal Almeida, who is either 26 or might be 27.

Figure 5. Sample Bad Output Generated by System

5. Conclusion

In this paper, we have presented a deep learning-based sentence fusion model. Automatic training data generation is done to produce a large amount of training data needed for training a text-to-text generative deep learning model. The performance is evaluated using test data created by human judges. The experimental results reveal that the model is effective in fusing sentences. We have a plan to apply this model for an abstract summary generation task.

References

1. **Conference**:Liao, K., Lebanoff, L., Liu, F.: Abstract meaning representation for multi-document summarization, In: Proceedings of the 27th International Conference on Computational Linguistics, COLING, Santa Fe, New Mexico, USA, pp. 1178–1190.(2018).
2. **Conference** Chenal, V., Cheung, J.C.K.: Predicting sentential semantic compatibility for aggregation in text-to-text generation, In 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, Osaka, Japan, pp. 1016-1070. (2016).
3. **Journal article:** Barzilay, R., McKeown, K.R.: Sentence fusion for multidocument news summarization, Computer Linguist. vol. 31 (3), pp. 297–328, (2005).

4. **Journal article:** Durrett, G., Berg-Kirkpatrick T., Klein, D.: Learning-based single-document summarization with compression and anaphoricity constraints", Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Berlin, Germany, Vol. 1, pp.1998–2008. August,(2016).
5. **Conference** Bing, L. et al.: Abstractive multi-document summarization via phrase selection and merging, In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Beijing, China, Vol. 1, pp.1587–1597.(2015).
6. **Journal article:** Martins, A.F.T., Smith, N.A.: Summarization with a joint model for sentence extraction and compression, In: Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing, In ILP, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 1–9.(2009).
7. **Journal article:** Chen Y., Bansal, M.: Fast abstractive summarization with reinforce-selected sentence rewriting, In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, vol. 1, pp. 675–686.(2018).
8. **Journal article:** Mendes, M. et al.: Jointly extracting and compressing documents with summary state representations, In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1, pp. 3955–3966.(2019).
9. **Conference** Thadani, K., McKeown, K.: Sentence Compression with Joint Structural Inference, In: Proceedings of the Seventeenth Conference on Computational Natural Language Learning, Sofia, Bulgaria, pp. 65-74.(2013).
10. **Workshop**:Marsi, E., Krahmer, E.: Classification of Semantic Relations by Humans and Machines, In: Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment, Association for Computational Linguistics, pp. 1-6.(2005).
11. **Journal article:** Filippova, K., Strube, M.: Dependency Tree Based Sentence Compression, In: Proceedings of the Fifth International Natural Language Generation Conference, Association for Computational Linguistics, pp. 25-32.(2008).
12. **Conference** Cheung, J., Penn, G.: Unsupervised Sentence Enhancement for Automatic Summarization, In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, Association for Computational Linguistics, pp. 775-786.(2014).
13. **Conference** Gerani, S. et al: Abstractive Summarization of Product Reviews Using Discourse Structure, In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP),Doha, Qatar, Association for Computational Linguistics, pp. 1602-1613,(2014).
14. **Journal article:** Mehdad, Y., Carenini, G., Tompa, F.W., NG, T.R.: Abstractive Meeting Summarization with Entailment and Fusion, In: Proceedings of the 14th European Work-shop on Natural Language Generation, Sofia, Bulgaria, Association for Computational Linguistics, pp. 136-146.(2013).
15. **Conference** Liu, F. et al: Toward Abstractive Summarization Using Semantic Representations, In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, "Denver, Colorado,Association for Computational Linguistics, pp. 1077-1086.(2015).
16. **Conference** Nayeem, M., Fuad, T., Chali, Y.: Abstractive Unsupervised Multi-Document Summarization using Paraphrastic Sentence Fusion, In: Proceedings of the 27th International Conference on Computational Linguistics, Santa Fe, New Mexico, USA, Association for Computational Linguistics",pp. 1191-1204.(2018).
17. **Workshop**:Lebanoff, L. et al.: Analyzing Sentence Fusion in Abstractive Summarization, In: Proceedings of the 2nd Workshop on New Frontiers in Summarization,Hong Kong, China,Association for Computational Linguistics", pp. 104-110.(2019).
18. **Journal article:** Erkan, G., Radev, D. R.: LexRank: Graph-Based Lexical Centrality as Salience in Text Summarization, AI Access Foundation, Vol. 22(1), pp. 457-479.(2004).
19. **Journal article:** Cao, Z., Wei, F., Li, W., Li, S.: Faithful to the original: Fact aware neural abstractive summarization, In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), (2018).
20. **Conference** Song, K., Zhao, L., Liu, F.: Structure-infused copy mechanisms for abstractive summarization, In: Proceedings of the International Conference on Computational Linguistics (COLING),(2018).
21. **Journal article:** See, A., Liu, P.J, Manning, C.D.: Get to the point: Summarization with pointer-generator networks, In: Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), (2017).
22. **Journal article:** Celikyilmaz, A., Bosselut, A., Xiaodong, H., Yejin, C.: Deep Communicating Agents for Abstractive Summarization, In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1, pp. 1662-1675.(2018).
23. **Journal article:** Raffel, C. et al.: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer,vol. 21,(2022).
24. **Journal article:** Lewis, M.: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension, In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, pp. 7871-7880.(2020).
25. **Journal article:** Jiwei, T., Xiaojun, W., Jianguo, X.: Abstractive document summarization with a graph-based attentional neural model, In: Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL),(2017).
26. **Conference** Gehrmann, S., Deng, Y., Rush, A.: Bottom-Up Abstractive Summarization, In: Proceedings of the 2018 Conference on Empirical

Methods in Natural Language Processing, Association for Computational Linguistics, pp. 4098-4109.(2018).
27. **Journal article:** Chen, Y., Bansal, M.: Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting, pp. 675-686.(2018).
28. **Journal article:** Lebanoff, L. , Song, K., Liu, F.: Adapting the Neural Encoder-Decoder Frame-work from Single to Multi-Document Summarization, In :Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, pp. 4131-4141.(2018).
29. **Conference:**Yang, L., Lapata, L.: Text Summarization with Pretrained Encoders, In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, 2019, Hong Kong, China, Association for Computational Linguistics, pp. 3728-3738.(2019).
30. **Conference** Callison, C., Osborne, M., Koehn, P.: Re-evaluating the role of BLEU in machine translation research, In: 11th conference of the european chapter of the association for computational linguistics, pp. 249-256. (2006).
31. **Journal article:** Papineni, K., Roukos, S., Ward, T., Zhu, W.Z.: 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, pp. 311-318.(2002).
32. **Journal article:** Lin, C.: ROUGE: A Package for Automatic Evaluation of Summaries, In: Text Summarization Branches Out, Association for Computational Linguistics, pp. 74-81.(2004).
33. **Journal article:** Wan X., Yang, J.: Improved Affinity Graph Based Multi-Document Summarization, In: Proceedings of the Human Language Technology Conference of the NAACL, Association for Computational Linguistics, pp. 181-184. (2006).
34. **Journal article:** Zolotareva, E., Tashu, T.M., Horv´ath, T.: Abstractive Text Summarization using Transfer Learning,(2020).
35. **Journal article:** Fatih, E., Guven, F., Galip, A.: Turkish abstractive text document summarization using text to text transfer transformer, Alexandria Engineering Journal, vol. 68, pp. 1-13.(2023).
36. **Journal article:** Sarkar, K.: Syntactic trimming of extracted sentences for improving extractive multi-document summarization, Journal of Computing, vol. 2, pp. 177-184.(2010).