CS7.401. Intro to NLP

Project Final Submission

Group Name: RantProMax

Topic: Text summarization specialized for news headline generation.

Member list:

Roll no.	Name
2022201006	Jaffrey Joy
2022201013	Amit Marathe
2022201046	Atharva Vaidya

Literature read:

- Yuning Mao et al., WWW '20: Proceedings of The Web Conference 2020. Pages 1773–1784. https://doi.org/10.1145/3366423.3380247
- Karolina Owczarzak and Hoa Trang Dang. 2011. Overview of the TAC 2011 summarization track: Guided task and AESOP task. In Proceedings of the Text Analysis Conference (TAC 2011).
- Over Paul and Yen James. 2004. An introduction to duc-2004. In Proceedings of the 4th Document Understanding Conference (DUC 2004).
- Alexander Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. 2019. Multi-News: A Large-Scale Multi-Document Summarization Dataset and Abstractive Hierarchical Model. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Pages 1074–1084. https://doi.org/10.18653/v1/P19-1102
- Masum KM, Abujar S, Tusher RTH, Faisal F, Hossai SA (2019) Sentence similarity measurement for Bengali abstractive text summarization. In: 2019 10th international conference on computing, communication and networking technologies (ICCCNT), Kanpur, India, 2019, pp 1–5. https://doi.org/10.1109/ICCCNT45670.2019.8944571
- Hanunggul PM, Suyanto S (2019) The impact of local attention in LSTM for abstractive text summarization. In: 2019 international seminar on research of information technology and intelligent systems (ISRITI), Yogyakarta, Indonesia, 2019, pp 54–57. https://doi.org/10.1109/ISRITI48646.2019.9034616
- Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks
- Luong M-T, Pham H, Manning CD (2015) Effective approaches to attention-based neural machine translation
- See A, Liu PJ, Manning CD (2017) Get to the point: summarization with pointer-generator networks
- Mohammad Masum K, Abujar S, Islam Talukder MA, Azad Rabby AKMS, Hossain SA (2019) Abstractive method of text summarization with sequence to sequence RNNs. In: 2019 10th international conference on computing, communication and networking technologies (ICCCNT), Kanpur, India, 2019, pp 1–5. https://doi.org/10.1109/ICCCNT45670.2019.8944620
- Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need
- The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)

Data Preprocessing

If there is an abundance of redundant or irrelevant information or data that is noisy and untrustworthy, the process of discovering knowledge during the training phase becomes more challenging. Examining data that has not been properly screened for these issues can result in inaccurate findings. Therefore, the quality and representation of the data should be given top priority before beginning work on any model.

We experimented with using stemming techniques such as Porter and Snowball from the nltk library but found them to produce a lesser presence in the glove word embeddings. Hence, we chose not to perform stemming on the data.

We also experimented with first performing word tagging using spaCy's Named Entity recognition module first and then cleaning the data using regex and vice versa and found that the former produced a better percentage of vocabulary presence in the glove embeddings and went ahead with that order.

After tagging we used the glove word embeddings to get a vectorized version of the dataset and used it for building the language model.

To enable faster processing using batch we have limited the headline length to 20 tokens and article length to 58 tokens.

The Models

1. Seq2Seq

The Seq2Seq model comprises an Encoder responsible for creating a context rich representation of the input sequence and providing it to the decoder. The decoder then produces a headline word-by-word.

It is a modification of the encoder–decoder model, the encoder responsible for creating context–rich vector re presentations from the news articles provided. The decoder generates the headline word by word, as the model calculates the attention given to the encoder representations at every instant.

The Encoder: It consists of a trainable embedding layer for which the Glove 100-dimensional word embedding is used. The recurrent layer is chosen to be a GRU (Gated Recurrent Units) layer of input size 100 which corresponds to the dimension of the vectors. The hidden size is chosen to be 256. The encoder processes the article in one instant.

- The input has shape [sequence, batch]
- After the embedding layer, we have the dimensions of [sequence, batch, 100]
- After the GRU layer, we have outputs of size [sequence, batch, 256] and the final hidden state of size [1, batch, 256] the layers is not bi-directional and has 1 layer

The decoder then generates the output sequence one word at a time, conditioned on the context vector and the previously generated words.

2. Seq2Seq with Attention

A seq2seq model with attention is a neural network architecture used for natural language processing tasks such as machine translation, text summarization, and dialogue generation. The architecture consists of two main components: an encoder and a decoder.

The encoder takes in the input sequence and produces a fixed-length vector representation of the sequence, which contains all the information necessary for generating the output sequence. This vector is called the "context vector."

The decoder then generates the output sequence one word at a time, conditioned on the context vector and the previously generated words. The attention mechanism is used to help the decoder focus on the most relevant parts of the input sequence at each step.

The Attentive Decoder: Every word of the headline $(y_1 \dots y_t)$ is generated one-by-one in the decoder as the attention mechanism calculates the attention over the encoder inputs. At every instant of the decoder for generating the next word, the significance e_{jt}

(the importance of the j^{th} word of the headline on the t^{th} time-step of the decoder) is calculated using the hidden states of the encoder h_t and the previous state of the decoder s_{t-1}.

These attention weights are calculated by linear layers, named time-step attention, and input attention corresponding to the hidden state of the decoder at the $t-1^{th}$ time step and the encoder outputs, respectively. The scalars αs that scales the encoder outputs are calculated and a SoftMax ensures that the distribution weighs the encoder inputs correctly. These are then concatenated with the previously generated word's embedding and fed to the GRU cell. The attention is implemented according to the following equation:

$$e_{jt} = V_{att}^{T}\left(\tanh\left(U_{att}s_{t-1} + W_{att}h_j\right)\right)$$

3. Transformer

The Transformer is a neural network architecture used in natural language processing (NLP) tasks. It uses self-attention to selectively focus on different parts of the input sequence when making predictions. The input sequence is first embedded into a high-dimensional vector space, and then a series of self-attention layers are applied to the embeddings. After the self-attention layers, the outputs are fed through a feedforward neural network, and the process is repeated for a fixed number of times. The final output is used for the specific NLP task at hand.

This contrasts with traditional recurrent neural networks (RNNs) that process the input sequence sequentially, which can lead to computational inefficiencies and difficulties in capturing long-term dependencies.

Architecture of the models

1. Attention

The attention model comprises of many neural network layers. These layers are explained in detail below-

1. **Embedding Layer**: This layer takes the one-hot encoded input tokens and converts them into dense vector representations that capture semantic meaning of words. In this model, pre-trained word embeddings are used, where each word is represented by a fixed-length vector that is learned from a large corpus of text data. This helps the model capture important semantic features of the input data, which are then used in later layers.
2. **Time-Step Attention Layer**: This layer is responsible for calculating the attention weight for the current decoder hidden state based on its similarity to the encoder hidden states at each time step. It takes as input the concatenated previous decoder hidden state and context vector, and applies a linear transformation followed by a non-linear activation function (ELU) to obtain a new representation of the hidden state. This representation is then used to compute the attention weight for each encoder hidden state using a dot product.
3. **Input-Sequence Attention Layer**: This layer is like the time-step attention layer, but instead of attending to the encoder hidden states at each time step, it attends to the hidden states for the entire input sequence. It takes as input the encoder hidden states for each time step, and applies a linear transformation followed by a non-linear activation function (ELU) to obtain a new representation of the encoder hidden states. This representation is then used in the time-step attention layer to compute the attention weights.
4. **Alpha Attention Layer**: This layer takes as input the concatenated representation of the encoder hidden states and the time-step attention hidden state and applies a linear transformation to obtain a scalar value representing the attention weight for the corresponding encoder hidden state. The attention dimension signifies the number of dimensions in the hidden state used to compute the attention weight, in this case 2*attention dim.
5. **Decoder GRU Layer**: This layer updates the hidden state of the decoder recurrently using the current input token and context vector obtained through attention mechanism. It takes as input the concatenated representation of the embedded input and the attention-based context vector and applies a GRU layer with dropout regularization to generate the next hidden state.
6. **Output Layer**: This layer takes as input the decoder hidden state and generates a probability distribution over the vocabulary using a linear transformation followed by a log-SoftMax activation. The output of this layer represents the probability of each token in the vocabulary being the next token in the generated headline.

2. Transformer

The attention model comprises of many neural network layers. These layers are explained in detail below-

1. **Embedding Layer**: This layer takes the input sequence and target sequence as input and converts them into dense embeddings of size embedding_dim. It uses the pre-trained embedding weights matrix to map each word in the sequence to a dense vector in the embedding space.
2. **Positional Encoding Layer**: This layer adds positional encoding to the input and target embeddings. It adds sinusoidal encoding of different frequencies and phases to the embeddings to provide the transformer model with information about the relative positions of the tokens in the sequence.
3. **Transformer Layer**: This layer consists of a stack of n transformer layers, each with two sub-layers:
    a. **Multi-Head Attention Sub-Layer**: This sub-layer takes the input and target embeddings as input and applies self-attention and multi-head attention mechanisms to capture the dependencies between the tokens in the input and target sequences.
    b. **Feedforward Sub-Layer:** This sub-layer applies a feedforward neural network to the output of the multi-head attention sub-layer.
4. **Output Layer**: This layer takes the output of the transformer layer and produces a vocabulary-sized output. It consists of a linear layer that maps the output of the transformer layer to the vocabulary size.

Results and Analysis:

ROGUE SCORE-

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a family of evaluation metrics used to evaluate the quality of automatic summarization and machine translation systems. ROUGE computes the overlap between the system-generated text and the reference summaries using various measures such as n-gram overlap, skip-grams, and word sequences.

ATTENTION:

Evaluation metrics	Precision	Recall	F1-Score
ROGUE-1 SCORE	0.2	0.20	0.2
ROGUE-L SCORE	0.15	0.13	0.14

TRANSFORMER:

Evaluation metrics	Precision	Recall	F1-Score
ROGUE-1 SCORE	0.52	0.32	0.4
ROGUE-L SCORE	0.51	0.31	0.39

On observing and analyzing the evaluation metrics, we can make the following comparative analysis between the attention model and the transformer model:

- Precision: The transformer model outperforms the attention model with a score of 0.52 compared to 0.20, indicating that the transformer model produces fewer false positives.
- Recall: The transformer model has a higher recall score of 0.32 compared to 0.13 for the attention model, meaning that the transformer model retrieves more relevant information.
- F1-Score: The transformer model has a higher F1-score of 0.4 compared to 0.2 for the attention model, which is a weighted average of the precision and recall, indicating that the transformer model performs better overall.
- ROGUE-1 Score: The transformer model has a higher ROGUE-1 score of 0.51 compared to 0.15 for the attention model, indicating that the transformer model generates more accurate summaries based on the unigram overlap between the generated summary and the reference summary.
- ROGUE-L Score: The transformer model has a higher ROGUE-L score of 0.39 compared to 0.14 for the attention model, indicating that the transformer model generates more accurate summaries based on the longest common subsequence between the generated summary and the reference summary.

Overall, the transformer model outperforms the attention model in all of the evaluation metrics provided. The transformer model has higher precision, recall, F1-score, and ROGUE scores, indicating that it is a better model for generating accurate summaries.

BLEU SCORE-

The BLEU (Bilingual Evaluation Understudy) score is a metric used to evaluate the quality of machine-translated text against one or more human reference translations. BLEU score is based on the n-gram overlap between the machine-generated text and the reference text. BLEU score ranges from 0 to 1, where a score closer to 1 indicates a better match between the generated text and the reference text.

ATTENTION:

```
BLEU SCORE: 0.32
```

TRANSFORMER:

```
BLEU SCORE: 0.58
```

Based on the BLEU score, we can make the following comparative analysis between the attention model and the transformer model:

- BLEU Score: The transformer model outperforms the attention model with a BLEU score of 0.58 compared to 0.32. BLEU (Bilingual Evaluation Understudy) is a metric used to evaluate the quality of machine translation output based on how well the generated text matches one or more reference translations.
- Comparative analysis: The transformer model generates output that is closer to the reference translations compared to the attention model, which suggests that the transformer model produces more accurate and fluent translations.

Overall, the transformer model outperforms the attention model on the BLEU metric score, with a higher score indicating better performance. This suggests that the transformer model is better at producing translations that are more like reference translations compared to the attention model.

COSINE SIMILARITY-

Cosine similarity is a measure of similarity between two non-zero vectors in a space that measures the cosine of the angle between them. In natural language processing, cosine similarity is often used to compute the similarity between two documents, or two sentences represented as vectors of word embeddings.

ATTENTION:

COSINE SIMILARITY:0.52

TRANSFORMER:

COSINE SIMILARITY:0.67

Based on observing and analyzing cosine similarity metric, we can make the following comparative analysis between the attention model and the transformer model:

- Cosine Similarity: The transformer model outperforms the attention model with a cosine similarity score of 0.67 compared to 0.52. The cosine similarity measures the similarity between the input and output vectors of the models, and the higher the score, the better the model is at generating a similar output to the input.
- Comparative analysis: The transformer model generates output that is more like the input compared to the attention model. This indicates that the transformer model is better able to capture the meaning of the input and produce output that preserves its semantic content.

Overall, the transformer model outperforms the attention model on the cosine similarity metric, with a higher score indicating better performance. This suggests that the transformer model is better at producing outputs that are similar in meaning to the input compared to the attention model.