Dank Learning: Generating Memes Using Deep Neural Networks

Abel L. Peirson V, E.Meltem Tolunay

Presenters:
Siba Smarak Panigrahi & Sohan Patnaik

Reading Session II
Kharagpur Data Analytics Group
IIT Kharagpur

February 28, 2021

Important Notes

- We expect you all to have certain queries regarding the presentation, certain intricate doubts maybe... Please put them in the **chat of this meeting**. We will address them all at the end!
- To ensure smooth flow of the presentation, we need you all to **keep your microphones and video turned off**.
- At the end, a **Google Form [Link] will be released** for feedback -BUT- most importantly, we ask you to put up name of **any AI-related paper** to be presented in the upcoming sessions!

Remember, if we can convert it into a presentation, we are indeed gonna present it!

Outline

Figure: Have a look at these memes!

Definition of Meme:
It has a definition? Yes, it has!

Why should we care?
I dunno, cause it's there?

A meme is an idea, behavior, or style that spreads from person to person within a culture often with the aim of conveying a particular phenomenon, theme, or meaning represented by the meme blah... blah... blah...

Some Output Memes!

Figure: Have a look at these memes!

Definition of Meme:
It has a definition? Yes, it has!

Why should we care?
I dunno, cause it's there?

A meme is an idea, behavior, or style that spreads from person to person within a culture often with the aim of conveying a particular phenomenon, theme, or meaning represented by the meme blah... blah... blah...

Some Output Memes!

Figure: Have a look at these memes!

Definition of Meme:
It has a definition? Yes, it has!

Why should we care?
I dunno, cause it's there?

A meme is an idea, behavior, or style that spreads from person to person within a culture often with the aim of conveying a particular phenomenon, theme, or meaning represented by the meme blah... blah... blah...

Brief Introductory Ideas

Image Captioning Model

Here are a few complicated stuff (to consume slides) - which would obviously be described later... duh...

- Image Captioning Models are nothing but simple RNN models that generate a caption for an image.
- This is a slightly changed Language Modelling Task.
- Okay, so before going to Language Modelling, we should be familiar with **Encoders** and **Decoders**.
- Encoder is nothing but a stack of several recurrent units where each accepts a single element of the input sequence, collects information for that element and propagates it forward.

Note: Don't worry, we will explain this when we come to explaining RNN architecures.

Image Captioning Model

Here are a few complicated stuff (to consume slides) - which would obviously be described later... duh...

- Image Captioning Models are nothing but simple RNN models that generate a caption for an image.
- This is a slightly changed Language Modelling Task.
- Okay, so before going to Language Modelling, we should be familiar with **Encoders** and **Decoders**.
- Encoder is nothing but a stack of several recurrent units where each accepts a single element of the input sequence, collects information for that element and propagates it forward.

Note: Don't worry, we will explain this when we come to explaining RNN architecures.

Image Captioning Model

Here are a few complicated stuff (to consume slides) - which would obviously be described later... duh...

- Image Captioning Models are nothing but simple RNN models that generate a caption for an image.
- This is a slightly changed Language Modelling Task.
- Okay, so before going to Language Modelling, we should be familiar with **Encoders** and **Decoders**.
- Encoder is nothing but a stack of several recurrent units where each accepts a single element of the input sequence, collects information for that element and propagates it forward.

Note: Don't worry, we will explain this when we come to explaining RNN architecures.

Image Captioning Model

Here are a few complicated stuff (to consume slides) - which would obviously be described later... duh...

- Image Captioning Models are nothing but simple RNN models that generate a caption for an image.
- This is a slightly changed Language Modelling Task.
- Okay, so before going to Language Modelling, we should be familiar with **Encoders** and **Decoders**.
- Encoder is nothing but a stack of several recurrent units where each accepts a single element of the input sequence, collects information for that element and propagates it forward.

Note: Don't worry, we will explain this when we come to explaining RNN architecures.

Image Captioning Model

Here are a few complicated stuff (to consume slides) - which would obviously be described later... duh...

- Image Captioning Models are nothing but simple RNN models that generate a caption for an image.
- This is a slightly changed Language Modelling Task.
- Okay, so before going to Language Modelling, we should be familiar with **Encoders** and **Decoders**.
- Encoder is nothing but a stack of several recurrent units where each accepts a single element of the input sequence, collects information for that element and propagates it forward.

Note: Don't worry, we will explain this when we come to explaining RNN architecures.

Image Captioning Model

Here are a few complicated stuff (to consume slides) - which would obviously be described later... duh...

- Image Captioning Models are nothing but simple RNN models that generate a caption for an image.
- This is a slightly changed Language Modelling Task.
- Okay, so before going to Language Modelling, we should be familiar with **Encoders** and **Decoders**.
- Encoder is nothing but a stack of several recurrent units where each accepts a single element of the input sequence, collects information for that element and propagates it forward.

Note: Don't worry, we will explain this when we come to explaining RNN architecures.

Image Captioning Model

- Dude, wait! **That sounds complicated!**
  Well, simply speaking you give an input to the network, it computes certain vector representation and propagates it forward. We will have a look at it again.
- Similarly, Decoder is a stack of several recurrent units where each unit predicts an output y_t at a time step t.

Note: Don't worry, we will explain this when we come to explaining RNN architecures.

Image Captioning Model

- Dude, wait! **That sounds complicated!**
  Well, simply speaking you give an input to the network, it computes certain vector representation and propagates it forward. We will have a look at it again.
- Similarly, Decoder is a stack of several recurrent units where each unit predicts an output y_t at a time step t.

Note: Don't worry, we will explain this when we come to explaining RNN architecures.

Image Captioning Model

- Dude, wait! **That sounds complicated!**
  Well, simply speaking you give an input to the network, it computes certain vector representation and propagates it forward. We will have a look at it again.
- Similarly, Decoder is a stack of several recurrent units where each unit predicts an output y_t at a time step t.

Note: Don't worry, we will explain this when we come to explaining RNN architecures.

RNNs for LMTs

- **RNNs**: Recurrent Neural Networks
  - Some high-fi stuff, if you do not know - IGNORE!
  - Well, I will try to explain with a hand-diagram. We will see, what happens.
- **LMTs**: Language Modelling Tasks
  - Any problem which find the following:

$$P(w_1, w_2, \cdots, w_n) : \text{Probability of this sequence of words.}$$

$$\text{Or equivalently } \; P(w_n | w_1, w_2, \cdots, w_{n-2}, w_{n-1})$$

RNNs for LMTs

- **RNNs**: Recurrent Neural Networks
  - Some high-fi stuff, if you do not know - IGNORE!
  - Well, I will try to explain with a hand-diagram. We will see, what happens.
- **LMTs**: Language Modelling Tasks
  - Any problem which find the following:

$$P(w_1, w_2, \cdots, w_n) : \text{Probability of this sequence of words.}$$

$$\text{Or equivalently } \; P(w_n | w_1, w_2, \cdots, w_{n-2}, w_{n-1})$$

RNNs for LMTs

- **RNNs**: Recurrent Neural Networks
  - Some high-fi stuff, if you do not know - IGNORE!
  - Well, I will try to explain with a hand-diagram. We will see, what happens.
- **LMTs**: Language Modelling Tasks
  - Any problem which find the following:

$$P(w_1, w_2, \cdots, w_n) : \text{Probability of this sequence of words.}$$

$$\text{Or equivalently } \; P(w_n | w_1, w_2, \cdots, w_{n-2}, w_{n-1})$$

RNNs for LMTs

- **RNNs**: Recurrent Neural Networks
  - Some high-fi stuff, if you do not know - IGNORE!
  - Well, I will try to explain with a hand-diagram. We will see, what happens.
- **LMTs**: Language Modelling Tasks
  - Any problem which find the following:

$$P(w_1, w_2, \cdots, w_n) \text{ : Probability of this sequence of words.}$$

Or equivalently $P(w_n | w_1, w_2, \cdots, w_{n-2}, w_{n-1})$

RNNs for LMTs

- **RNNs**: Recurrent Neural Networks
  - Some high-fi stuff, if you do not know - IGNORE!
  - Well, I will try to explain with a hand-diagram. We will see, what happens.
- **LMTs**: Language Modelling Tasks
  - Any problem which find the following:

$$P(w_1, w_2, \cdots, w_n) : \text{Probability of this sequence of words.}$$

$$\text{Or equivalently } P(w_n | w_1, w_2, \cdots, w_{n-2}, w_{n-1})$$

RNNs for LMTs

- **RNNs**: Recurrent Neural Networks
  - Some high-fi stuff, if you do not know - IGNORE!
  - Well, I will try to explain with a hand-diagram. We will see, what happens.
- **LMTs**: Language Modelling Tasks
  - Any problem which find the following:

$$P(w_1, w_2, \cdots, w_n) : \text{Probability of this sequence of words.}$$

$$\text{Or equivalently } \; P(w_n | w_1, w_2, \cdots, w_{n-2}, w_{n-1})$$

Some over the mind equations to make the slide attractive:

Inside RNN unit:

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1})$$

$$f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1})$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1})$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{cx}x_t + W_{cm}m_{t-1})$$

$$m_t = o_t \odot c_t$$

$$p_{t+1} = softmax(m_t)$$

Click Here to see a wonderful explanation on LSTMs!

GloVe and Attention

- Well, you got to know about word embeddings which are indeed the inputs to the RNN units.
- **GloVe** i.e. Global Vectors are vector representations of words pretrained for use in NLP tasks.
- Note that the vector is 300 dimensional.
- Now, let us know about Attention.
- **Attention Mechanism** is an approach of that enables to highlight relevant features of the input data essential in the prediction of output sequence/words.

GloVe and Attention

- Well, you got to know about word embeddings which are indeed the inputs to the RNN units.
- **GloVe** i.e. Global Vectors are vector representations of words pretrained for use in NLP tasks.
- Note that the vector is 300 dimensional.
- Now, let us know about Attention.
- **Attention Mechanism** is an approach of that enables to highlight relevant features of the input data essential in the prediction of output sequence/words.

GloVe and Attention

- Well, you got to know about word embeddings which are indeed the inputs to the RNN units.
- **GloVe** i.e. Global Vectors are vector representations of words pretrained for use in NLP tasks.
- Note that the vector is 300 dimensional.
- Now, let us know about Attention.
- **Attention Mechanism** is an approach of that enables to highlight relevant features of the input data essential in the prediction of output sequence/words.

GloVe and Attention

- Well, you got to know about word embeddings which are indeed the inputs to the RNN units.
- **GloVe** i.e. Global Vectors are vector representations of words pretrained for use in NLP tasks.
- Note that the vector is 300 dimensional.
- Now, let us know about Attention.
- **Attention Mechanism** is an approach of that enables to highlight relevant features of the input data essential in the prediction of output sequence/words.

Data Preprocessing

- The dataset consists of approximately 400,000 image, label and caption triplets with 2600 unique image-label pairs.
- **Labels:** They are short descriptions referring to the image, i.e. the meme template, and are the same for identical images.
- Accordingly, each **image-label pair** is associated with several (roughly 160) different **captions**.
- **Preprocessing**:
  - Each word in the caption was lowercased
  - Punctuation marks were assigned their own tokens
  - Start, end and unknown tokens were introduced into the vocabulary with randomly initialized word vectors.
  - Every word that appears fewer than 3 times in the corpus is set to UNK. Captions with more than 2 UNKs are removed from the dataset.

Data Preprocessing

- The dataset consists of approximately 400,000 image, label and caption triplets with 2600 unique image-label pairs.
- **Labels:** They are short descriptions referring to the image, i.e. the meme template, and are the same for identical images.
- Accordingly, each **image-label pair** is associated with several (roughly 160) different **captions**.
- **Preprocessing**:
  - Each word in the caption was lowercased
  - Punctuation marks were assigned their own tokens
  - Start, end and unknown tokens were introduced into the vocabulary with randomly initialized word vectors.
  - Every word that appears fewer than 3 times in the corpus is set to UNK. Captions with more than 2 UNKs are removed from the dataset.

Data Preprocessing

- The dataset consists of approximately 400,000 image, label and caption triplets with 2600 unique image-label pairs.
- **Labels:** They are short descriptions referring to the image, i.e. the meme template, and are the same for identical images.
- Accordingly, each **image-label pair** is associated with several (roughly 160) different **captions**.
- **Preprocessing**:
  - Each word in the caption was lowercased
  - Punctuation marks were assigned their own tokens
  - Start, end and unknown tokens were introduced into the vocabulary with randomly initialized word vectors.
  - Every word that appears fewer than 3 times in the corpus is set to UNK. Captions with more than 2 UNKs are removed from the dataset.

Data Preprocessing

- The dataset consists of approximately 400,000 image, label and caption triplets with 2600 unique image-label pairs.
- **Labels:** They are short descriptions referring to the image, i.e. the meme template, and are the same for identical images.
- Accordingly, each **image-label pair** is associated with several (roughly 160) different **captions**.
- **Preprocessing**:
  - Each word in the caption was lowercased
  - Punctuation marks were assigned their own tokens
  - Start, end and unknown tokens were introduced into the vocabulary with randomly initialized word vectors.
  - Every word that appears fewer than 3 times in the corpus is set to UNK. Captions with more than 2 UNKs are removed from the dataset.

Data Preprocessing

- The dataset consists of approximately 400,000 image, label and caption triplets with 2600 unique image-label pairs.
- **Labels:** They are short descriptions referring to the image, i.e. the meme template, and are the same for identical images.
- Accordingly, each **image-label pair** is associated with several (roughly 160) different **captions**.
- **Preprocessing**:
  - Each word in the caption was lowercased
  - Punctuation marks were assigned their own tokens
  - Start, end and unknown tokens were introduced into the vocabulary with randomly initialized word vectors.
  - Every word that appears fewer than 3 times in the corpus is set to UNK. Captions with more than 2 UNKs are removed from the dataset.

Data Preprocessing

- The dataset consists of approximately 400,000 image, label and caption triplets with 2600 unique image-label pairs.
- **Labels:** They are short descriptions referring to the image, i.e. the meme template, and are the same for identical images.
- Accordingly, each **image-label pair** is associated with several (roughly 160) different **captions**.
- **Preprocessing**:
  - Each word in the caption was lowercased
  - Punctuation marks were assigned their own tokens
  - Start, end and unknown tokens were introduced into the vocabulary with randomly initialized word vectors.
  - Every word that appears fewer than 3 times in the corpus is set to UNK. Captions with more than 2 UNKs are removed from the dataset.

Data Preprocessing

- The dataset consists of approximately 400,000 image, label and caption triplets with 2600 unique image-label pairs.
- **Labels:** They are short descriptions referring to the image, i.e. the meme template, and are the same for identical images.
- Accordingly, each **image-label pair** is associated with several (roughly 160) different **captions**.
- **Preprocessing**:
  - Each word in the caption was lowercased
  - Punctuation marks were assigned their own tokens
  - Start, end and unknown tokens were introduced into the vocabulary with randomly initialized word vectors.
  - Every word that appears fewer than 3 times in the corpus is set to UNK. Captions with more than 2 UNKs are removed from the dataset.

Model Architecture: Encoder

- Encoder is used to provide a meaningful initial state to the decoder to initiate the text generation process. To capture the image embeddings, **Inceptionv3** model, pretrained on the ILSVRC-2012-CLS image classification dataset.
- The model outputs a 2048 dimensional vector (this is not equal to word embedding space that is 300 dimensional)
- Hence, project the image embeddings into the word embedding space using a trainable fully connected layer.
- The paper uses three different types of encoders - we explain them briefly.

Model Architecture: Encoder

- Encoder is used to provide a meaningful initial state to the decoder to initiate the text generation process. To capture the image embeddings, **Inceptionv3** model, pretrained on the ILSVRC-2012-CLS image classification dataset.
- The model outputs a 2048 dimensional vector (this is not equal to word embedding space that is 300 dimensional)
- Hence, project the image embeddings into the word embedding space using a trainable fully connected layer.
- The paper uses three different types of encoders - we explain them briefly.

Model Architecture: Encoder

- Encoder is used to provide a meaningful initial state to the decoder to initiate the text generation process. To capture the image embeddings, **Inceptionv3** model, pretrained on the ILSVRC-2012-CLS image classification dataset.
- The model outputs a 2048 dimensional vector (this is not equal to word embedding space that is 300 dimensional)
- Hence, project the image embeddings into the word embedding space using a trainable fully connected layer.
- The paper uses three different types of encoders - we explain them briefly.

Model Architecture: Encoder

- Encoder is used to provide a meaningful initial state to the decoder to initiate the text generation process. To capture the image embeddings, **Inceptionv3** model, pretrained on the ILSVRC-2012-CLS image classification dataset.
- The model outputs a 2048 dimensional vector (this is not equal to word embedding space that is 300 dimensional)
- Hence, project the image embeddings into the word embedding space using a trainable fully connected layer.
- The paper uses three different types of encoders - we explain them briefly.

Encoder I

- Just use the meme templates and disregard the labels completely.
- The inputs to the decoder solely include encoding from the images.
- Only an image is required to generate a meme.
- Let p $\in \mathbb{R}^{2048}$ be the inception output corresponding to a meme template, and let q $\in \mathbb{R}^{300}$ be the decoder initial state

$$q = W_1 p + b_1$$

Encoder I

- Just use the meme templates and disregard the labels completely.
- The inputs to the decoder solely include encoding from the images.
- Only an image is required to generate a meme.
- Let $p \in \mathbb{R}^{2048}$ be the inception output corresponding to a meme template, and let $q \in \mathbb{R}^{300}$ be the decoder initial state

$$q = W_1 p + b_1$$

Encoder I

- Just use the meme templates and disregard the labels completely.
- The inputs to the decoder solely include encoding from the images.
- Only an image is required to generate a meme.
- Let p $\in \mathbb{R}^{2048}$ be the inception output corresponding to a meme template, and let q $\in \mathbb{R}^{300}$ be the decoder initial state

$$q = W_1 p + b_1$$

Encoder I

- Just use the meme templates and disregard the labels completely.
- The inputs to the decoder solely include encoding from the images.
- Only an image is required to generate a meme.
- Let p $\in \mathbb{R}^{2048}$ be the inception output corresponding to a meme template, and let q $\in \mathbb{R}^{300}$ be the decoder initial state

$$q = W_1 p + b_1$$

Encoder II

- Obtain the **image embeddings** (output of Inceptionv3)
- Obtain the **pretrained GloVe embedding** for each word present in the meme label and compute their average.
- Concat this averaged vector to the image embedding.
- Feed the concatenated vector into a trainable fully connected layer.
- The output of the fully connected layer is fed into the decoder.
- Let $e_i \in \mathbb{R}^{300}$ represent the GloVe embeddings of the label words

$$q = W_2(p || \frac{e_1 + ... + e_n}{n}) + b_2$$

Encoder II

- Obtain the **image embeddings** (output of Inceptionv3)
- Obtain the **pretrained GloVe embedding** for each word present in the meme label and compute their average.
- Concat this averaged vector to the image embedding.
- Feed the concatenated vector into a trainable fully connected layer.
- The output of the fully connected layer is fed into the decoder.
- Let $e_i \in \mathbb{R}^{300}$ represent the GloVe embeddings of the label words

$$q = W_2(p || \frac{e_1 + \ldots + e_n}{n}) + b_2$$

Encoder II

- Obtain the **image embeddings** (output of Inceptionv3)
- Obtain the **pretrained GloVe embedding** for each word present in the meme label and compute their average.
- Concat this averaged vector to the image embedding.
- Feed the concatenated vector into a trainable fully connected layer.
- The output of the fully connected layer is fed into the decoder.
- Let $e_i \in \mathbb{R}^{300}$ represent the GloVe embeddings of the label words

$$q = W_2(p||\frac{e_1 + \dots + e_n}{n}) + b_2$$

Encoder II

- Obtain the **image embeddings** (output of Inceptionv3)
- Obtain the **pretrained GloVe embedding** for each word present in the meme label and compute their average.
- Concat this averaged vector to the image embedding.
- Feed the concatenated vector into a trainable fully connected layer.
- The output of the fully connected layer is fed into the decoder.
- Let $e_i \in \mathbb{R}^{300}$ represent the GloVe embeddings of the label words

$$q = W_2(p || \frac{e_1 + ... + e_n}{n}) + b_2$$

Encoder II

- Obtain the **image embeddings** (output of Inceptionv3)
- Obtain the **pretrained GloVe embedding** for each word present in the meme label and compute their average.
- Concat this averaged vector to the image embedding.
- Feed the concatenated vector into a trainable fully connected layer.
- The output of the fully connected layer is fed into the decoder.
- Let $e_i \in \mathbb{R}^{300}$ represent the GloVe embeddings of the label words

$$q = W_2(p || \frac{e_1 + ... + e_n}{n}) + b_2$$

Encoder II

- Obtain the **image embeddings** (output of Inceptionv3)
- Obtain the **pretrained GloVe embedding** for each word present in the meme label and compute their average.
- Concat this averaged vector to the image embedding.
- Feed the concatenated vector into a trainable fully connected layer.
- The output of the fully connected layer is fed into the decoder.
- Let $e_i \in \mathbb{R}^{300}$ represent the GloVe embeddings of the label words

$$q = W_2(p||\frac{e_1 + ... + e_n}{n}) + b_2$$

Encoder II: Image

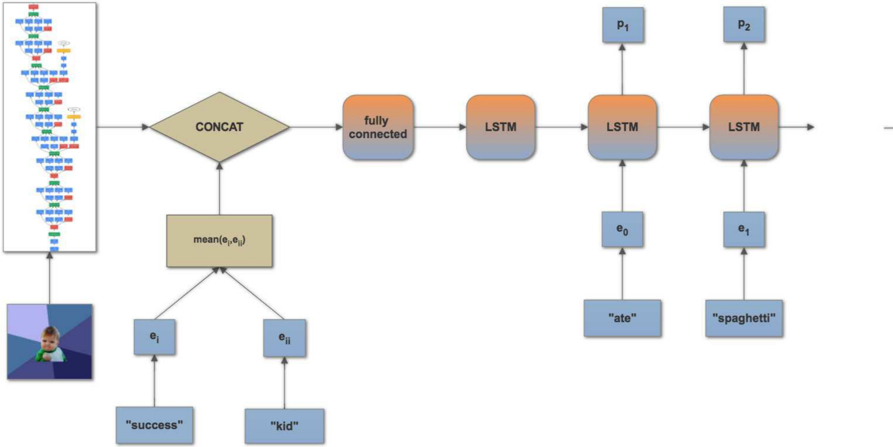

Figure: Encoder II

Encoder III

- Obtain the image embeddings and put them through a fully connected layer
- Extend the encoder with an additional LSTM network.
- This network takes the projected image embedding as the initial state and runs the GloVe embeddings of the labels through the LSTM.
- Perform attention on the encoder LSTM cells (while finding the decoder states) using **Luong attention mechanism** [Link].
- The output of this additional LSTM network serves as the initial state of the decoder.

$$x_t = W_3 p + b_3, q = m_t$$

Encoder III

- Obtain the image embeddings and put them through a fully connected layer
- Extend the encoder with an additional LSTM network.
- This network takes the projected image embedding as the initial state and runs the GloVe embeddings of the labels through the LSTM.
- Perform attention on the encoder LSTM cells (while finding the decoder states) using **Luong attention mechanism** [Link].
- The output of this additional LSTM network serves as the initial state of the decoder.

$$x_t = W_3 p + b_3, q = m_t$$

Encoder III

- Obtain the image embeddings and put them through a fully connected layer
- Extend the encoder with an additional LSTM network.
- This network takes the projected image embedding as the initial state and runs the GloVe embeddings of the labels through the LSTM.
- Perform attention on the encoder LSTM cells (while finding the decoder states) using **Luong attention mechanism** [Link].
- The output of this additional LSTM network serves as the initial state of the decoder.

$$x_t = W_3 p + b_3, q = m_t$$

Encoder III

- Obtain the image embeddings and put them through a fully connected layer
- Extend the encoder with an additional LSTM network.
- This network takes the projected image embedding as the initial state and runs the GloVe embeddings of the labels through the LSTM.
- Perform attention on the encoder LSTM cells (while finding the decoder states) using **Luong attention mechanism** [Link].
- The output of this additional LSTM network serves as the initial state of the decoder.

$$x_t = W_3 p + b_3, q = m_t$$

Encoder III

- Obtain the image embeddings and put them through a fully connected layer
- Extend the encoder with an additional LSTM network.
- This network takes the projected image embedding as the initial state and runs the GloVe embeddings of the labels through the LSTM.
- Perform attention on the encoder LSTM cells (while finding the decoder states) using **Luong attention mechanism** [Link].
- The output of this additional LSTM network serves as the initial state of the decoder.

$$x_t = W_3 p + b_3, q = m_t$$

Model Architecture: Decoder

- The decoder consists of a unidirectional LSTM network that operates according to the 6 equations described previously.
- The authors did one modification i.e. they introduced the use of pretrained GloVe embeddings rather than randomly initialised word vectors.
- Note that the word embedding vectors were left trainable but the advantage of using pre-trained vectors was that the training time will be reduced to a great extent as the values are already near to the optimal state.

Model Architecture: Decoder

- The decoder consists of a unidirectional LSTM network that operates according to the 6 equations described previously.
- The authors did one modification i.e. they introduced the use of pretrained GloVe embeddings rather than randomly initialised word vectors.
- Note that the word embedding vectors were left trainable but the advantage of using pre-trained vectors was that the training time will be reduced to a great extent as the values are already near to the optimal state.

Model Architecture: Decoder

- The decoder consists of a unidirectional LSTM network that operates according to the 6 equations described previously.
- The authors did one modification i.e. they introduced the use of pretrained GloVe embeddings rather than randomly initialised word vectors.
- Note that the word embedding vectors were left trainable but the advantage of using pre-trained vectors was that the training time will be reduced to a great extent as the values are already near to the optimal state.

Model Architecture: Decoder

- As mentioned earlier, we have 3 types encoders. So for the first two encoders, a simple decoder is employed where the output of the encoder is used as the input to the decoder to generate the meme caption.
- For the third encoder, Attention Mechanism is used for the decoder outputs to capture the content more precisely.

Model Architecture: Decoder

- As mentioned earlier, we have 3 types encoders. So for the first two encoders, a simple decoder is employed where the output of the encoder is used as the input to the decoder to generate the meme caption.
- For the third encoder, Attention Mechanism is used for the decoder outputs to capture the content more precisely.

Beam Search Decoding

- The decoder outputs at each time step are nothing but probabilities for the occurrence of a particular word.
- Greedy decoding simply picks the most probable word which might prove to be bad as a caption and provide less humorous, not so random outputs.
- To get better results, the authors used **Beam Search Decoding**, where k most probable words are chosen, and then those are used to further generate the words until we reach the end of the sentence.
- Note that k is a hyper-parameter that is tuned during training.

Beam Search Decoding

- The decoder outputs at each time step are nothing but probabilities for the occurrence of a particular word.
- Greedy decoding simply picks the most probable word which might prove to be bad as a caption and provide less humorous, not so random outputs.
- To get better results, the authors used **Beam Search Decoding**, where k most probable words are chosen, and then those are used to further generate the words until we reach the end of the sentence.
- Note that k is a hyper-parameter that is tuned during training.

Beam Search Decoding

- The decoder outputs at each time step are nothing but probabilities for the occurrence of a particular word.
- Greedy decoding simply picks the most probable word which might prove to be bad as a caption and provide less humorous, not so random outputs.
- To get better results, the authors used **Beam Search Decoding**, where k most probable words are chosen, and then those are used to further generate the words until we reach the end of the sentence.
- Note that k is a hyper-parameter that is tuned during training.

Beam Search Decoding

- The decoder outputs at each time step are nothing but probabilities for the occurrence of a particular word.
- Greedy decoding simply picks the most probable word which might prove to be bad as a caption and provide less humorous, not so random outputs.
- To get better results, the authors used **Beam Search Decoding**, where k most probable words are chosen, and then those are used to further generate the words until we reach the end of the sentence.
- Note that k is a hyper-parameter that is tuned during training.

Temperature Function

- One more variant is used i.e. the Temperature Function.
- 100 most probable words are chosen from the decoder outputs and then relative probabilities are computed among these 100 words before choosing the k words amongst them.

$$f(p)_i = \frac{p_i^{1/T}}{\sum_j p_j^{1/T}}$$

- Note that $T = 1$ corresponds to unchanged probabilities, high T leads to a very flat distribution and low T yields the inefficient Greedy Decoding

Training

- Each model variant were trained using 1, 2 and 3 layered versions of the LSTM decoder network.
- **Momentum** and **SGD** optimizers were used for learning the weights. Hyperparameters were thouroughly tuned to find the best learning rate schedule, batch size and LSTM/attention unit size.
- Evaluation metric used was **perplexity** which we will come in a while.
- No significant change was observed in perplexity score when 2 and 3 LSTM layers were used in the decoder network instead of 1.

Training

- Each model variant were trained using 1, 2 and 3 layered versions of the LSTM decoder network.
- **Momentum** and **SGD** optimizers were used for learning the weights. Hyperparameters were thouroughly tuned to find the best learning rate schedule, batch size and LSTM/attention unit size.
- Evaluation metric used was **perplexity** which we will come in a while.
- No significant change was observed in perplexity score when 2 and 3 LSTM layers were used in the decoder network instead of 1.

Training

- Each model variant were trained using 1, 2 and 3 layered versions of the LSTM decoder network.
- **Momentum** and **SGD** optimizers were used for learning the weights. Hyperparameters were thouroughly tuned to find the best learning rate schedule, batch size and LSTM/attention unit size.
- Evaluation metric used was **perplexity** which we will come in a while.
- No significant change was observed in perplexity score when 2 and 3 LSTM layers were used in the decoder network instead of 1.

Training

- Each model variant were trained using 1, 2 and 3 layered versions of the LSTM decoder network.
- **Momentum** and **SGD** optimizers were used for learning the weights. Hyperparameters were thouroughly tuned to find the best learning rate schedule, batch size and LSTM/attention unit size.
- Evaluation metric used was **perplexity** which we will come in a while.
- No significant change was observed in perplexity score when 2 and 3 LSTM layers were used in the decoder network instead of 1.

Network Performance

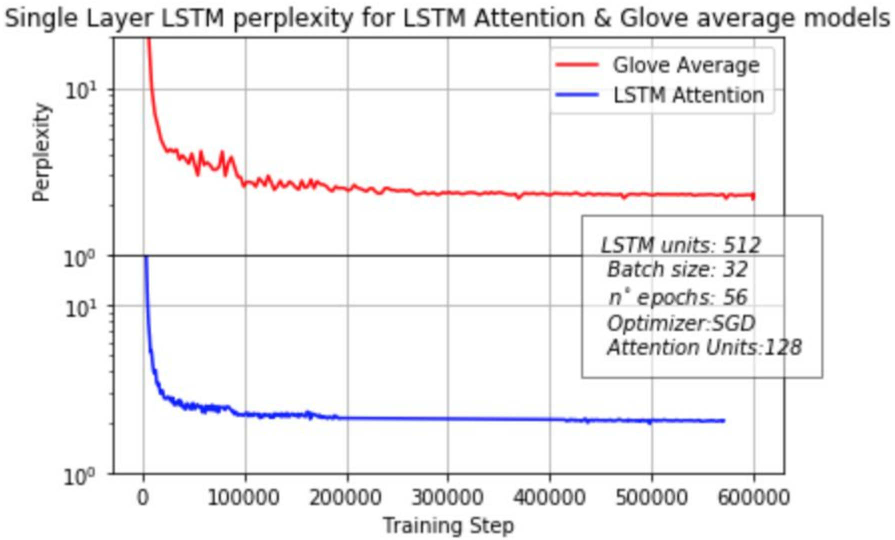

Figure: Performance

Results and Evaluation

- **Perplexity** (PP) is a measure of the inverse probabilities of predicting the next word in the example caption (C)

$$PP(C) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1, \cdots, w_{i-1})}}$$

Here, $w_1, \cdots, w_N$ are the words in caption C.

- Low perplexity [**less confused(perplexed) model**] tells us how well the model is learning to caption images of different formats with the correct style.
- But this is a limited metric since it tells nothing about the hilarity of captions and whether they are original and varied.
- The authors show 20 different memes to five people to test the differentiability & hilarity of generated memes and original ones: they were nearer to being indistinguishable and near hilarity to original memes (of the dataset)

Results and Evaluation

- **Perplexity** (PP) is a measure of the inverse probabilities of predicting the next word in the example caption (C)

$$PP(C) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1, \cdots, w_{i-1})}}$$

  Here, $w_1, \cdots, w_N$ are the words in caption C.
- Low perplexity [**less confused(perplexed) model**] tells us how well the model is learning to caption images of different formats with the correct style.
- But this is a limited metric since it tells nothing about the hilarity of captions and whether they are original and varied.
- The authors show 20 different memes to five people to test the differentiability & hilarity of generated memes and original ones: they were nearer to being indistinguishable and near hilarity to original memes (of the dataset)

Results and Evaluation

- **Perplexity** (PP) is a measure of the inverse probabilities of predicting the next word in the example caption (C)

$$PP(C) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1, \cdots, w_{i-1})}}$$

  Here, $w_1, \cdots, w_N$ are the words in caption C.
- Low perplexity [**less confused(perplexed) model**] tells us how well the model is learning to caption images of different formats with the correct style.
- But this is a limited metric since it tells nothing about the hilarity of captions and whether they are original and varied.
- The authors show 20 different memes to five people to test the differentiability & hilarity of generated memes and original ones: they were nearer to being indistinguishable and near hilarity to original memes (of the dataset)

Results and Evaluation

- **Perplexity** (PP) is a measure of the inverse probabilities of predicting the next word in the example caption (C)

$$PP(C) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1, \cdots, w_{i-1})}}$$

  Here, $w_1, \cdots, w_N$ are the words in caption C.
- Low perplexity [**less confused(perplexed) model**] tells us how well the model is learning to caption images of different formats with the correct style.
- But this is a limited metric since it tells nothing about the hilarity of captions and whether they are original and varied.
- The authors show 20 different memes to five people to test the differentiability & hilarity of generated memes and original ones: they were nearer to being indistinguishable and near hilarity to original memes (of the dataset)

Conclusion

- This paper proposed how to generate meme captions using Neural Networks.
- Just to sum up everything, 3 different encoder schemes were employed with and without caption labels.
- LSTM network in the decoder was fine tuned for Language Modelling of humorous meme captions.

Thank You!

Yo! That's All For Spring Break :)

Thank You!
Yo! That's All For Spring Break :)