Paraphrase Generation

Adam Wiles
Northeastern University
wiles.a@husky.neu.edu

Abstract
The goal of this project is to build a model capable of paraphrase generation using a multi-layer LSTM network. Paraphrase research has mostly centered around thesaurus based or rule based approaches. Recently, there has been research done using LSTMs and a sequence-to-sequence approach, which has also been used for machine translation. This approach involves a similar technique but also adds more information, each word’s part of speech tag, with goal of improving paraphrases through hidden connections between tags.

1 Introduction
Paraphrasing is changing the syntax, diction, or word ordering of a sentence without changing the semantics. This can be viewed as a monolingual machine translation because the meaning of the sentence doesn’t change while the actual sentence does. This can be useful for QA systems in rewording found text or even rewording questions. It is also technically challenging. For these reasons this project tries to produce a model capable of paraphrasing sentences.

2 Related Work
2.1 Using a Deep LSTM Network
This research works using a stacked set of LSTM encoder and decoders [1]. This style of network is widely used in the field of NLP. Data is taken from three major data sets and represented in a low dimensional format before being trained on. The results of this work is both an improved paraphrase generator and a benchmark for further paraphrase generation on the three data sets used, one of which we intend to use.

2.2 Using VAE and LSTM Models
Building off of the work discussed in 1.2.1 this paper discusses introducing a deep generative component into the previously discussed LSTM model [2]. Specifically this work uses Variational Autoencoders (VAEs). This particular innovation is outside of our realm of knowledge, however the given results provide us another benchmark to compare our results to.

3 Data
The main problem in choosing a dataset for this problem was deciding what is considered a paraphrase. There is no real standard dataset when training a model for paraphrasing and depends on how long of text is desired. We decided to use Microsoft’s COCO dataset which offered a wide range of labeled sentence level paraphrases.

3.1 Microsoft Common Objects in Context (MSCOCO)
The MSCOCO dataset was created by having multiple unique participants look at an image and write down a short descriptions. Each image contains a bordered and highlighted object to focus the descriptions. Each of these descriptions is effectively a paraphrase of the other. An example of an image used for the dataset can be seen below.

The data was collected by first downloading the 2017 version of the COCO dataset. This contained a little over 180,000 image annotations and is split into train and test sets. The train set contains roughly 160,000 annotations and the test set 40,000.

4 Approach

With the dataset downloaded, each annotation was grouped by image and aligned with another sentence in it’s group. These sentences are considered paraphrases and serve as a source, target pairs for training.

| Two sheep are looking over a cement wall. | Two sheep peeking over the wall at something. |
| Two sheep poking their heads over a white cement wall | Two sheep are looking over a cement wall. |
| Two sheep peek over the edge of a wall. | Two sheep poking their heads over a white cement wall |
| Sheep standing next to a stone wall in front of a tree. | Two sheep peek over the edge of a wall. |
| Two sheep peeking over the wall at something. | Sheep standing next to a stone wall in front of a tree. |

Above shows an example pairing of sentences within their group to another sentence that is not the same sentence.

From here each sets vocabulary was pre-built and any words that occurred less than ten times was replaced with an unknown token, <UNK>. There was also a vocab for the tabs prebuilt using the python library NLTK’s, Natural Language Toolkit, tag library.

While the exact method and approach evolved through changes in the model (these changes are discussed in the next section) the system for setup remained roughly the same. First, the sentence
pairs of source and target are loaded. Then the sentences are tokenized and each tokens part of speech is determined. Then the sentence is filtered through the pre-built vocab to add <UNK>s to any word out of the vocabulary. The part of speech is determined first in order to preserve sentence tag structure even if the word is out of vocabulary. The system changes at this part depending on model but these sentences are combined into a vector and feed through the model. Depending on how the encoded output matches up to the encoded target the model weights are adjusted. This is done across the train dataset.

The idea behind this approach is that the more information about a sentence the more connections a model can make for paraphrase.

5 Experiments

This being the first attempt at building a neural network the idea was to add the part of speech information, but slowly build and evolve the model as the project continues and more was learned about neural networks. Each model was designed and created using TensorFlow.

5.1 Multilayer Perceptron
Due to a lack of familiarity with neural networks the initial model was created using a multilayered perceptron. The model was created using the approach described in section 4 with the vector created as below.

<table>
<thead>
<tr>
<th>w₁</th>
<th>t₁</th>
<th>w₂</th>
<th>t₂</th>
<th>...</th>
<th>wₘₐₓ</th>
<th>tₘₐₓ</th>
</tr>
</thead>
</table>

This figure shows the vector used with the first word of the sentence followed by the part of speech tag for that word. This was pattern was repeated with the second word then third word all the way to the length of the longest description in the data. If the sentence was shorter than this it was padded with zeros. This vector was then feed into a multi-layered perceptron with first a single hidden layer and then two hidden layers. This approach was not promising and decided to move on to the next design.

5.2 LSTM

More recent research has leveraged the power of Long Short Term Memory blocks for paraphrase generation and so this was chosen for the next design. Again, the approach for setup follows the description in section 4. This time, however, the length of the vector was limited to 20 words or 40 with tags and words. If a sentence was longer than 20 words it was removed. This length limit still included the majority of the descriptions and gave a better output vector. Again, if a sentence was less than 20 words in length it was padded with zeros.

<table>
<thead>
<tr>
<th>w₁</th>
<th>t₁</th>
<th>w₂</th>
<th>t₂</th>
<th>...</th>
<th>w₂₀</th>
<th>t₂₀</th>
</tr>
</thead>
</table>

This vector was feed to a series of stacked LSTMs which then produced an output vector representing the encoded paraphrase. The output vector is of length 20 and decoded for

5.3 Sequence-to-Sequence
This approach follows that described in Prakash et. al. [1] paper with the modifications of input. The model works by setting up an a vector similar to the ones described in previous models except with start and end tokens. A example of this vector can be seen below.
This vector is stacked into a batch of 32 and feed first through an encoder then to a sequence of LSTMs. These are then given to a decoder with is given a context state based on Bahdanau Attention and appropriately decoded.

6 Evaluation

There ended up being two evaluation methods for this task. First, being a kind of gold standard which just involved reading it and seeing if it made any sense especially compared to the annotations. For this task this was effective for giving a rough estimate but for an actually metric a BLEU score was used.

6.1 BLEU Score
BLEU score is a metric that is typically used in machine translation problems. Since, paraphrasing can be mapped to machine translation the BLEU score also proves a good measure for testing paraphrases. It works by giving a reference sentence and the paraphrased sentence. For this the target and output sentences were used, respectively. The score is then calculated by comparing the number of common n-grams used in the reference and paraphrased sentence. It then gives a score between 0 and 1. This score is often multiplied by 10 or 100.

6.2 Multilayer Perceptron
Due to the long limit on the input vector, the results of the perceptron were long and incomprehensible. Since, the original plan was to use LSTMs it was decided to move quickly to the LSTM and not bother evaluating with a BLEU score. Although it would likely have been very low.

6.3 LSTM
The LSTM output results which were incomprehensible but occasionally would show a couple of similar phrases at the beginning of a sentence. An example of output sentences can be seen below.

source: a small bathroom with a toilet by the window and a pedestal sink with a mirrored medicine cabinet over it

target: light restroom taking jumbo world in down closing empty design opening amongst bicycle somewhat commercial before older riding older plain

While the majority of the sentence makes no sense it seems like the phrase small bathroom and light restroom are similar. This would happen occasionally in the output of the model. When this is feed to a BLEU score it receives a score of 0. Based on these results it was decided to move to a approach more similar to machine translation.

6.4 Sequence-to-Sequence
The seq 2 seq results did much better but were limited because of time and computational power. An example output:

source: a small bathroom with a toilet by the window and a pedestal sink with a
mirrored medicine cabinet over it

target: light restroom taking jumbo world in down closing empty design opening amongst bicycle somewhat commercial before older riding older plain

The average BLEU score for this model was 14.56. This was a much better improvement but still below the BLEU score of the Prakash et al [1] at 36.7.

7 Conclusion

As the work here was completed it seems that adding part of speech tags does not improve any models ability generate paraphrases. However, there was limited time and computational power which keep the model from fully training and testing. With the ability to completely test and adjust things according I think there is promise in adding extra features to network.

References
