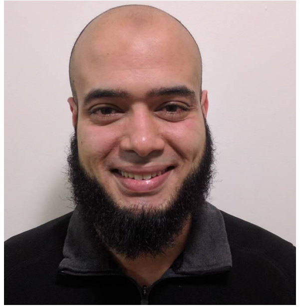
Machine Learning

RNN for Word Generation

Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / MSc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



Next Word Prediction

- In next character prediction, the vocabulary was simple
  - For example, 100 letters (for letters, puntcations, etc)
  - Pros: we use **one-hot-encoding** as input feature
  - Cons: harder to learn good output and need a long training
- In the next word prediction, the vocabulary can be like 100k words!
  - Cons: it is hard to keep providing inputs for 100k one-hot-encoding
    - Also, no relationship between close words!
      - Man/Woman. Apple/Banana. King/Queen
  - Solution: Instead, use **embeddings** (dense vector representations of words)
  - For example, the 100k words are represented using 100k x 64 data
    - We init them and learn them during the training itself
    - We may use the learned one in another app
- Relevant note: **Perplexity** is a common metric for evaluation for some NLP tasks

Word Embedding

- Technique in NLP where each **word** is **mapped** to a **feature vector**
- Goal: replacing the sparse one-hot encoded vectors with a dense compact representation where similar words have continuous representation
  - Hopefully, similar words has similar encoding.
- But how to find such representations?! Learn them
  - Decide a dimension for the word, e.g. 128
  - Create for K words K feature vectors and initialize randomly
  - Map each input index to its current vector and feed the vectors to the network
- Hence, the embeddings are learned and updated during the training process of the network, in the **same way** as other learnable parameters.
  - We call these embeddings vectors = Embedding matrix

nn.Embedding

- A PyTorch layer to convert categorical data (indices), into a learnable embedding space.
- The internal vectors are passed to the network and updated during the training
- Key Syntax
  - Creation: self.embedding = nn.Embedding(vocab_size=200000, embedding_dim=128)
    - Create 200k embeddings, each vector is 128 features
  - Train/Inference in forward function
    - Let x be a tensor of input indices (e.g. [2, 7, 10])
    - x = self.embedding(x)
      - Output x will be 3x128

```python
class WordRNN(nn.Module):
    def __init__(self, vocab_size, embedding_dim, hidden_dim, n_layers=1):
        super(WordRNN, self).__init__()
        self.hidden_dim = hidden_dim
        self.n_layers = n_layers
        self.rnn = nn.RNN(embedding_dim, hidden_dim, n_layers, batch_first=True)
        self.fc = nn.Linear(hidden_dim, vocab_size)

        self.embedding = nn.Embedding(vocab_size, embedding_dim)

    def forward(self, x, hidden = None):
        x = self.embedding(x) # convert idx to embedding_dim: words x embedding_dim

        if hidden is None:
            out, hidden = self.rnn(x)
        else:
            out, hidden = self.rnn(x, hidden)

        out = self.fc(out)
        return out, hidden
```

Next Word Prediction

- There is no code differences except using **an embedding layer**!
- Divide a string into words (tokens), this gives us the vocabulary size
  - Map a word to an index and the opposite
- Use word like EOS to mark the end of a sequence
- Code is attached
  - You will notice training is faster to reach high accuracy (overfitting a small dataset)
- Note: people either train and learn words embeddings or download a pretrained word embeddings (you may finetune).

ChatGpt

Custom instructions

What would you like ChatGPT to know about you to provide better responses?

Machine Learning
Computer Vision
Multiview 3D Geometry
Sensor Calibration
Self-driving
in-cabin

MO **You**

if x = 2 and y =3, what is x * y

ChatGPT

If $x = 2$ and $y = 3$, then the product $x \times y$ is $2 \times 3 = 6$.

MO **You**

And x+y?

ChatGPT

If $x = 2$ and $y = 3$, then the sum $x + y$ is $2 + 3 = 5$.

“Acquire knowledge and impart it to the people.”

“Seek knowledge from the Cradle to the Grave.”