Volume 11 Issue 2 (2022)

Natural Language Generation Using Machine Learning Techniques

Angelina Yang[1] and Sadaf Halim[#]

[1]Plano West Senior High School, Plano, TX, USA
[#]Advisor

ABSTRACT

Natural language generation is a subfield of machine learning, consisting of creating systems that can produce understandable texts in the human language. It is applied to all areas dealing with reporting and content creation, such as journalism and online chatbots. Despite natural language generation being labeled as a subfield, it covers a vast range of topics beyond the scope of this paper. Instead, this research paper aims to provide an overview on select topics within natural language generation: Word Embedding, Long Short-Term Memory (LSTM), and Encoder-Decoder Architecture. The authors have analyzed and reinterpreted so that the audience has an improved understanding of natural language generation in spite of the topic's broad reach.

Introduction

Machine Learning

Machine learning functions through automating decision-making processes by generalizing from known examples. One of the most prevalent types of machine learning is supervised learning. In supervised learning, the system is given pairs of inputs and outputs; with this data, the system must produce desired outputs given new inputs. The "supervised" aspect lies in said desired outputs for each learned example. Supervised learning performance is easy to measure, as the system's accuracy is presented as percent confidence. This form of machine learning is widely implemented in analyzing and sorting handwriting, online activity, and images. Note that most of these examples require following a general pattern, which is recognized and adjusted as the machine passes through more data. While there exists supervised learning, there also exists unsupervised learning, which is beyond the scope of this paper.

Neural Networks

Artificial neural network processes are inspired by biological neural networks. Just as how the human body transmits stimulus through interconnected neurons to the brain, triggering a reaction, artificial neural networks work in a similar fashion. A neural network is divided into layers. The input is transformed into bits or pixels, forming the nodes of the input layer. The output layer has as many nodes as the model's class labels. Each output will have a resulting probability that represents the output's confidence. The layers in between the input and output layers are hidden layers. There is not a solid number for how many hidden layers a neural network can have. Connecting these layers are "activations" from the previous layers that affect the "activations" in the following layers.

The significance of certain values to the output is determined by weights. These weights connect each node from the previous layer to every node in the following layer. The value of these weights affects the weighted sum of the previous layer's nodes after a series of calculations, which are beyond the scope of this paper. Weights are represented in numbers of both positive and negative values, and the exact values of these weights are indeterminable. This is due to weights being learned through feeding the network datasets. Initially, the first dataset will pass through weights of completely random values. By adding up the squares of differences in accuracy after the first processing, the network will slightly modify the weights' values through an algorithm called gradient descent, which aims to minimize the mean squared error. As more datasets are passed through the network, the randomized weights will become more specific.

Natural Language Generation

Natural language generation is a subfield of machine learning and computational linguistics. It comprises producing written narratives after feeding the system a variety of text datasets. These models are often used in content creation such as journalism and chatbots, producing fluent, stylized responses to customer prompts. Natural language generative models are also used in the complexities of language translation and summarizations. In the most basic sense, Natural language generation consists of adding words to the tail of a sentence based on a dataset's sentence patterns. The simplest natural language model will analyze bigrams, a pair of consecutive words, that occur most frequently and will make decision based on the previous word. However, this is a nearsighted approach to sentence generation and has a higher chance of producing a grammatically incorrect sentence. Thus, more complex natural language models are bidirectional and will analyze characters present both early and late into the sentence before making a decision.

Word Embedding

Word embedding creates numerical representations of words in vectors. It is important to transform words into numeric vectors since computers cannot recognize words. Numbers are assigned to words based on semantic and conceptual similarities; if three words are assigned close values and are used similarly in a sentence, then the model will interpret the words as similar as it generates new sentences. However, if numbers were randomly assigned to words, resulting in no pattern for the model to analyze, then the generated sentences will lose sense. When similar words are assigned distant numbers, it results in misclassifications. The semantic patterns categorized by numbers are used to determine positive (+1) and negative (-1) connotations, thus stressing the importance of similar semantics being assigned numbers close in value. For instance, if the word "accident" was assigned a value between the values assigned to "happy" and "kind", then the model will assume "accident" has a similarly positive connotation as the other two words.

Vectors usually contain more than one value to describe words, assigning a value to each characteristic of the word. Compared to using only one value per word, having multiple values will help better relate words in closeness for the model. The longer the vector representation of the word is, the greater flexibility there is in how the vector can define a word.

The similarity of words given their respective vectors are measured by notions of distance. Similarity is usually calculated using the Manhattan distance, which is the sum of absolute differences between two vectors. The resulting distance represents the frequency that the compared words appear in a body of text together. Due to this form of representation, some words such as "sun" and "moon" will have closer relations than "sun" and "sunshine".

Long Short-Term Memory

Long Short-Term Memory (LSTM) networks are capable of ordering objects according to recent and older data. At its core, LSTM in natural language generation utilizes both local and distant context to make informed decisions. Thus, such networks are known to be bidirectional, where information travels in both directions to successfully produce a fluent sentence.

In order to understand LSTM, it is important to also understand recurrent neural networks (RNN) because LSTM is a subgroup of RNN. In an RNN, an input passes through a recurrent neural network, a value is outputted, and then the information is looped back to influence the next output. Essentially, the information is continuously passed on to the next network — which is actually identical to the previous — to help create outputs. With this structure, RNNs are successful in prediction tasks such as language modeling and image captioning. The flaw in this, however, is that if the relevant information is too far from the present task, then the RNN would be unable to connect information and provide a proper output.

LSTM models were created to avoid the long-term dependency issue. The text input begins in its cell state. The input passes through a series of gates containing mathematical transformations, where information is added and removed to the cell. As the input passes through those gates, it is converted into its internal representation called a "hidden state" or "latent state". While in a latent state, the amount of transferred information is determined by a value between 0 and 1 calculated by a sigmoid layer. The resulting output is more specific to sentiment and thus more contextually accurate.

As to how this is conceptually applied to language models, LSTM processes are similar to using n-grams in choosing the next word. N-grams are an ordered sequence of n words given a sample text. For now, the exemplified language model will operate on a bigram, or a pair of consecutive words. As the language model adds words to the end of a sentence, it will find the bigram occurring most frequently in a dataset, where the last word in the sentence is the first word of the bigram. The obvious issue with this process is that it is nearsighted, having a higher chance of outputting a grammatically incorrect sentence. Using a trigram or a larger n-gram would result in something more likely to be grammatically correct. It is this concept that is applied by LSTM models.

Encoder-Decoder Architecture

Encoder-decoder architecture is a process where an input is encoded into a cryptic representation and then decoded into an output. The process involves the "attention" which places emphasis on a specific part of the input most relevant to the prediction step. The results are highly accurate machine translations, language modelings, and speech to text transcriptions. Despite having previously mentioned that LSTM could avoid the long-term dependency issue, encoder-decoder architecture actually has stronger back propagation skills and can access a bigger window of text to reference from.

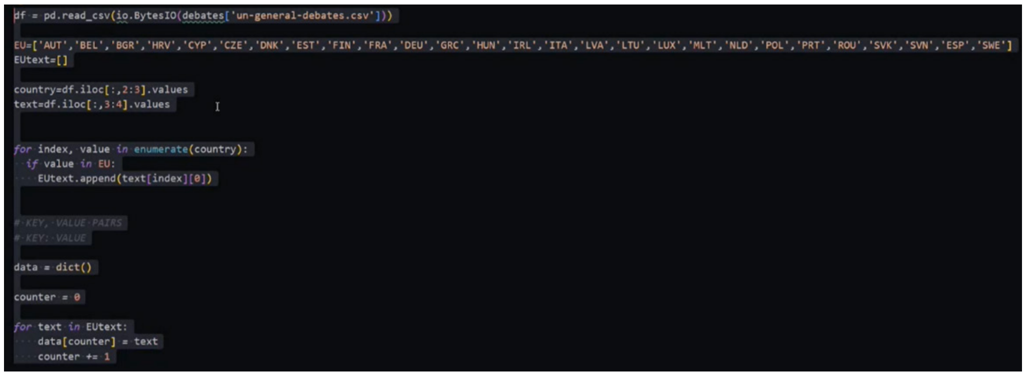
Given an input sentence, encoder-decoder architecture will first pass the input through a word embedding and gain a vector representation of the words. The vectors then pass through the encoder and transform into an abstract representation holding learned information. In the encoder layer, there exists a multi-headed attention module that computes attention weights for each word. The attention model relates words with other words. How closely each word is associated with another is determined in a score matrix; the higher the score, the higher the relation. Through mathematical transformations, the higher scores are heightened and the lower scored are reduced, helping the system become more confident on which words to attend to. Before the encoded representation is sent to the decoder, is it possible for it to pass through the encoder n times, each time learning a new attention representation.

The decoder has similar layers as the encoder and generates an output sequence given information from the encoder and a previous network, much like RNN —and simultaneously LSTM— models. It is autoregressive, taking past information to predict future information and stops when it generates a token. As similar to the encoder, the input from a previous network pass through an embedding layer and a multi-headed attention module. However, this multi-headed attention module has a more specific function attention between given words with words following behind it. It does so by masking, which only allows words to attend to itself and past words. The information from the encoder passes through a separate multi-headed attention module and is compared with the decoder's inputs to further determine relevant attention on which encoder inputs. Through further processing the information reaches the linear classifier which generates a predicted output. The predicted output is then fed back into the decoder over again until a token stops the process.

Application

This section explains a simple natural language generation model. This model seeks to create original speeches based on debate transcripts. In this application, Python was chosen to code this natural language model. It is a general-purpose programming language with easy use of domain-specific scripting languages. It has benefits such as being able to interact directly with the code using terminals and libraries for special-purpose functionalities. Example libraries used in the model include NumPy for advanced mathematical functions, pandas for data wrangling and analysis, and matplotlib for creating visualizations such as line charts and scatter plots.

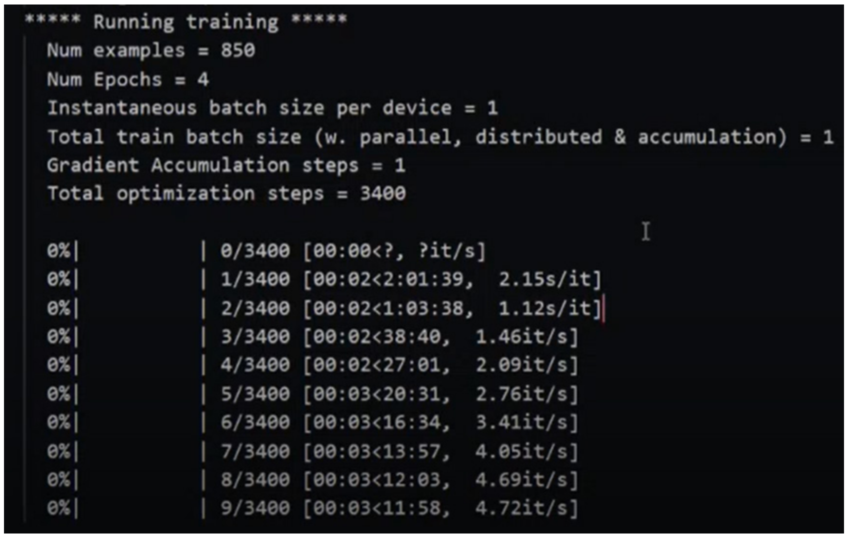
The dataset chosen for the natural language model is a series of general debates from 1970 to 2016. Some processing was done to the dataset so that the speeches were narrowed down to only countries belonging to a certain organization. The data is then put into a dictionary.



```
df = pd.read_csv(io.BytesIO(debates['un-general-debates.csv']))

EU=['AUT','BEL','BGR','HRV','CYP','CZE','DNK','EST','FIN','FRA','DEU','GRC','HUN','IRL','ITA','LVA','LTU','LUX','MLT','NLD','POL','PRT','ROU','SVK','SVN','ESP','SWE']
EUtext=[]

country=df.iloc[:,2:3].values
text=df.iloc[:,3:4].values


for index, value in enumerate(country):
  if value in EU:
    EUtext.append(text[index][0])


# KEY, VALUE PAIRS
# KEY: VALUE

data = dict()

counter = 0

for text in EUtext:
    data[counter] = text
    counter += 1
```

Figure 1. Preprocessing. The dataset is filtered using loops so that it only contains speeches from a certain alliance, specifying the results of the network.

The dataset is split into a training batch and a testing batch. The training batch is passed through the neural network in order to adjust the weights, and the test batch is later passed to test the accuracy of the network. Once the process completes, the program will be able to generate text through another set of code given a prompt. If given a starting phrase such as "The current financial situation is…", then the model will be able to add words and grammatical symbols to it based on patterns in the dataset, eventually producing a fluent, legible script.



```
***** Running training *****
  Num examples = 850
  Num Epochs = 4
  Instantaneous batch size per device = 1
  Total train batch size (w. parallel, distributed & accumulation) = 1
  Gradient Accumulation steps = 1
  Total optimization steps = 3400

  0%|          | 0/3400 [00:00<?, ?it/s]
  0%|          | 1/3400 [00:02<2:01:39,  2.15s/it]
  0%|          | 2/3400 [00:02<1:03:38,  1.12s/it]
  0%|          | 3/3400 [00:02<38:40,  1.46it/s]
  0%|          | 4/3400 [00:02<27:01,  2.09it/s]
  0%|          | 5/3400 [00:03<20:31,  2.76it/s]
  0%|          | 6/3400 [00:03<16:34,  3.41it/s]
  0%|          | 7/3400 [00:03<13:57,  4.05it/s]
  0%|          | 8/3400 [00:03<12:03,  4.69it/s]
  0%|          | 9/3400 [00:03<11:58,  4.72it/s]
```

Figure 2. Running training. The terminal displays the process of training batches being passed through the neural network.

Discussion

Developments in natural language generation significantly contribute to commercial and industrial artificial intelligence, creating chatbots, translators, and virtual assistants. Just as how LSTM networks succeeds RNN networks and encoder-decoder architecture succeeds LSTM, neural networks will continuously be augmented to overcome its shortcomings as it is applied. The more natural language generation is used in fields such as research and customer service, the more important it is for the knowledge of machine learning to spread. If enterprises and influential figures could better understand the concepts of natural language generation, they could gain higher public support and a more accurate understanding of their audience. Likewise, common people armed with the knowledge of natural language generation can make more educated decisions when encountering a site driven by neural networks. Regardless, natural language generation is a constantly updating field with still much to be explored.

Conclusion

Natural language generation is a type of machine learning, consisting of training using a text dataset and producing fluent sentences. As it is a type of machine learning, natural language generation models pass the data through neural networks, which assign weights that determine the significance of certain traits in a dataset. This data usually passes through a word embedding layer which transforms words into long vectors of numbers, each number representing a different characteristic. To produce fluent, legible sentences, the model will ask for a prompt of words. A Long short-term generation model will utilize local and distant context in the phrase to continue adding words to the sentence. Though, encoder-decoder architecture has comparatively better back propagation skills due to its use of attention, which emphasizes specific data that is most relevant to producing the best choice of words.

Limitations

Natural language generation is a massive topic, covering concepts, mathematics, and coding. Thus, the scope of the paper can be seen as a simplification in order to cover specific topics. A few sources used in this paper are dated back to 7 years ago, creating the possibility that some information is outdated by recent innovations.

Acknowledgments

I would like to thank Mr. Sadaf M.D. Halim for providing me guidance, suggestions, and brief concepts for this article.

References

3Blue1Brown. (n.d.). Retrieved August 17, 2022, from https://www.3blue1brown.com/topics/neural-networks

Breuel, T. (2015, August 11). Benchmarking of LSTM Networks.

Dale R (2020). Natural language generation: The commercial state of the art in 2020. Natural Language Engineering 26, 481–487. https://doi.org/10.1017/S135132492000025X

Dilmegani, C. (2020). Natural language generation (NLG): What it is & how it works. AIMultiple. Retrieved August 17, 2022, from https://research.aimultiple.com/nlg/

Gatt, A., & Krahmer, E. (2018). Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. Journal of Artificial Intelligence Research, 61, 65–170. https://doi.org/10.1613/jair.5477

McDonald, D. (1986). (rep.). Natural Language Generation. Amherst, MA: University of Massachusetts.

Mu¨ller Andreas Christoph, & Guido, S. (2016). Introduction to machine learning with python: A guide for data scientists. Oreilly et Associates Inc.

Olah, C. (2015). [web log]. Retrieved 2022, from https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Phi, M. (2020, April 27). Illustrated Guide to Transformers Neural Network: A step by step explanation.

Reiter, E. (n.d.). (rep.). Building Natural Language Generation Systems. University of Aberdeen.

Shekhar, S. (2021, June 30). LSTM for text classification: Beginners Guide to Text Classification. Analytics Vidhya. Retrieved August 17, 2022, from https://www.analyticsvidhya.com/blog/2021/06/lstm-for-text-classification/