How to generate poems according to specific values

Poetries are kind of text, and they are sequential data unlike images, because we can process one image inside a neural network at once, but we cannot process one poetry inside a neural network at once, because every poetry has a different count of words, thus we need to define a sequence length to process a sequence of words at once, thus we need to use Recurrent Neural Network which is widely use for generating text. We use word level method for this purpose to predict word by word unlike character level method which predicts character by character like TPU Shakespeare project by Google https://colab.research.google.com/github/tensorflow/tpu/blob/master/tools/colab/shakespeare_with_tpuestimator.ipynb

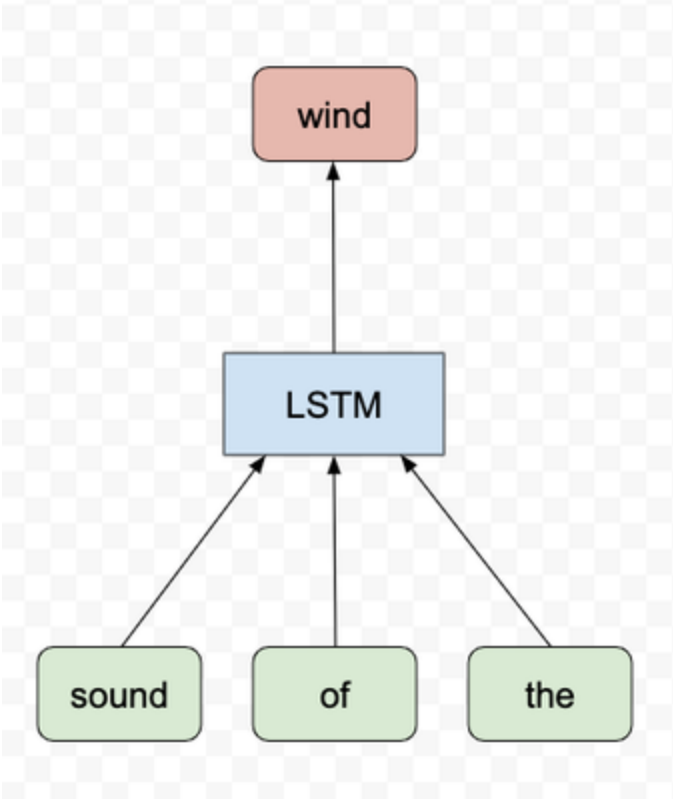
RNN takes a sequence of words of a length and predicts the next word, thus we created our training data from the poetries that we have as follows:

X Data			Y Data
The	sound	of	the
sound	of	the	wind

Our RNN-LSTM model consists of 3 layers

1- an input layer consists of 3 neurons
2- a hidden layer consists of one LSTM cell
3- an output layer consists of one neuron

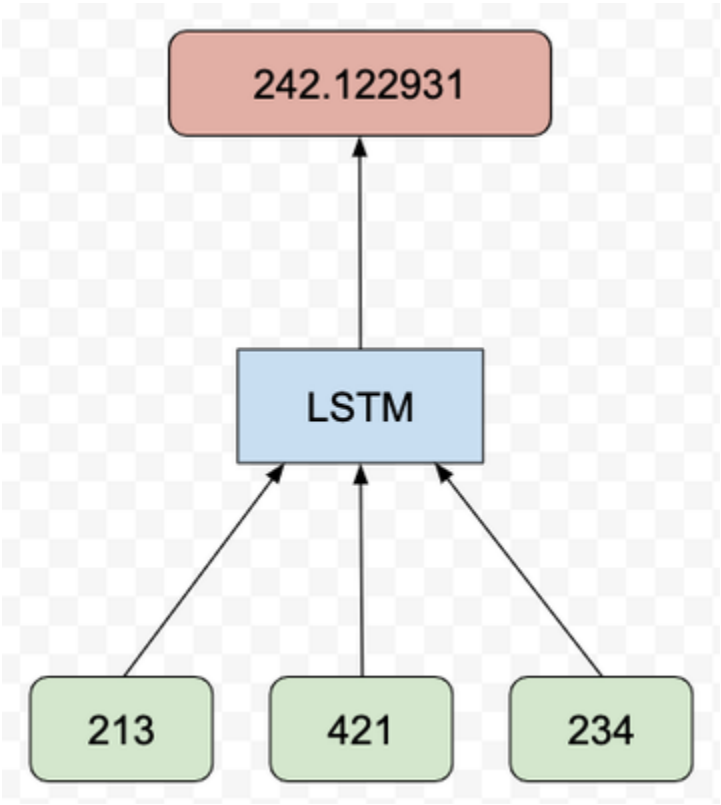
as in the following picture:



Building the word2id dictionary:
we need to assign every unique word that we have to a unique integer number as an index for all poetries that we have, like in the following picture:

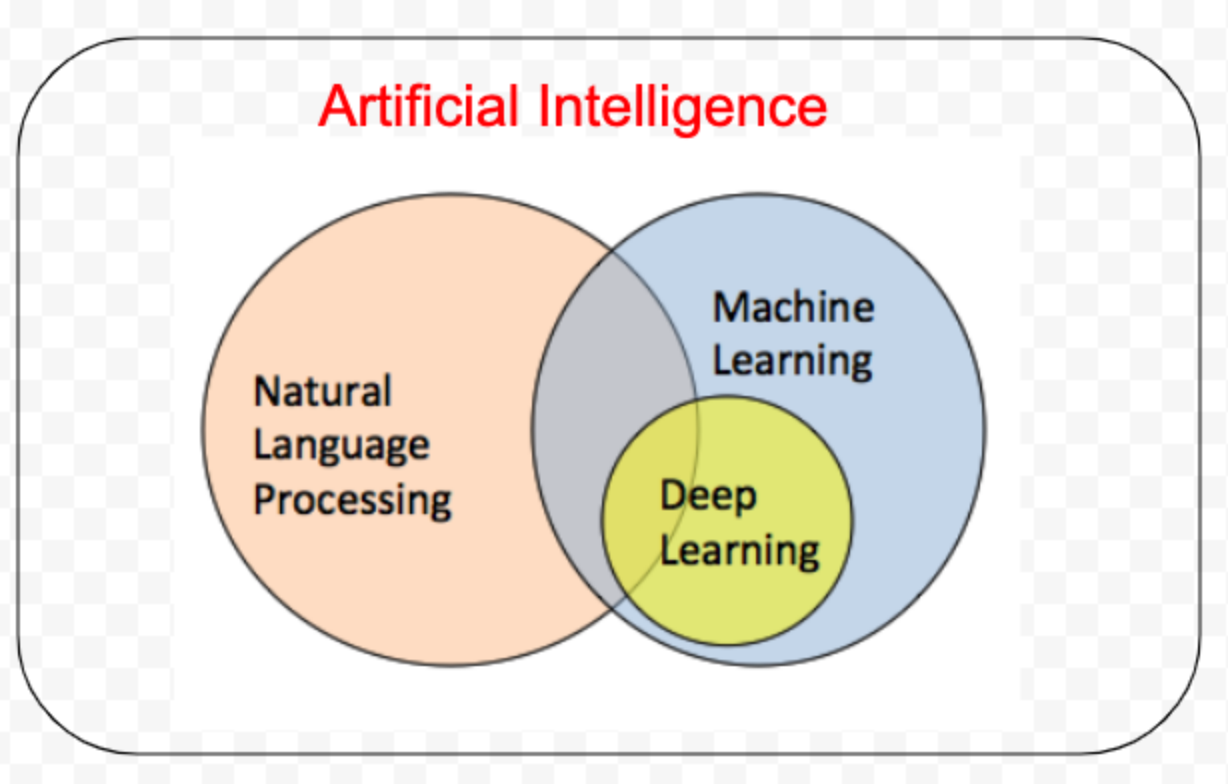
available	1600
such	270
clear	245
consistent	158
relevant	144
sufficient	138
strong	107

However, words themself are not the input for the RNN, we need to use their unique integers from the word2id dictionary that we've created, then the neural network will predict the integer index of the next word.



Unfortunately, the result was very bad, because the predicted values were only in a specific range and that what tried at the beginning.

Using another AI field: Natural Language Processing



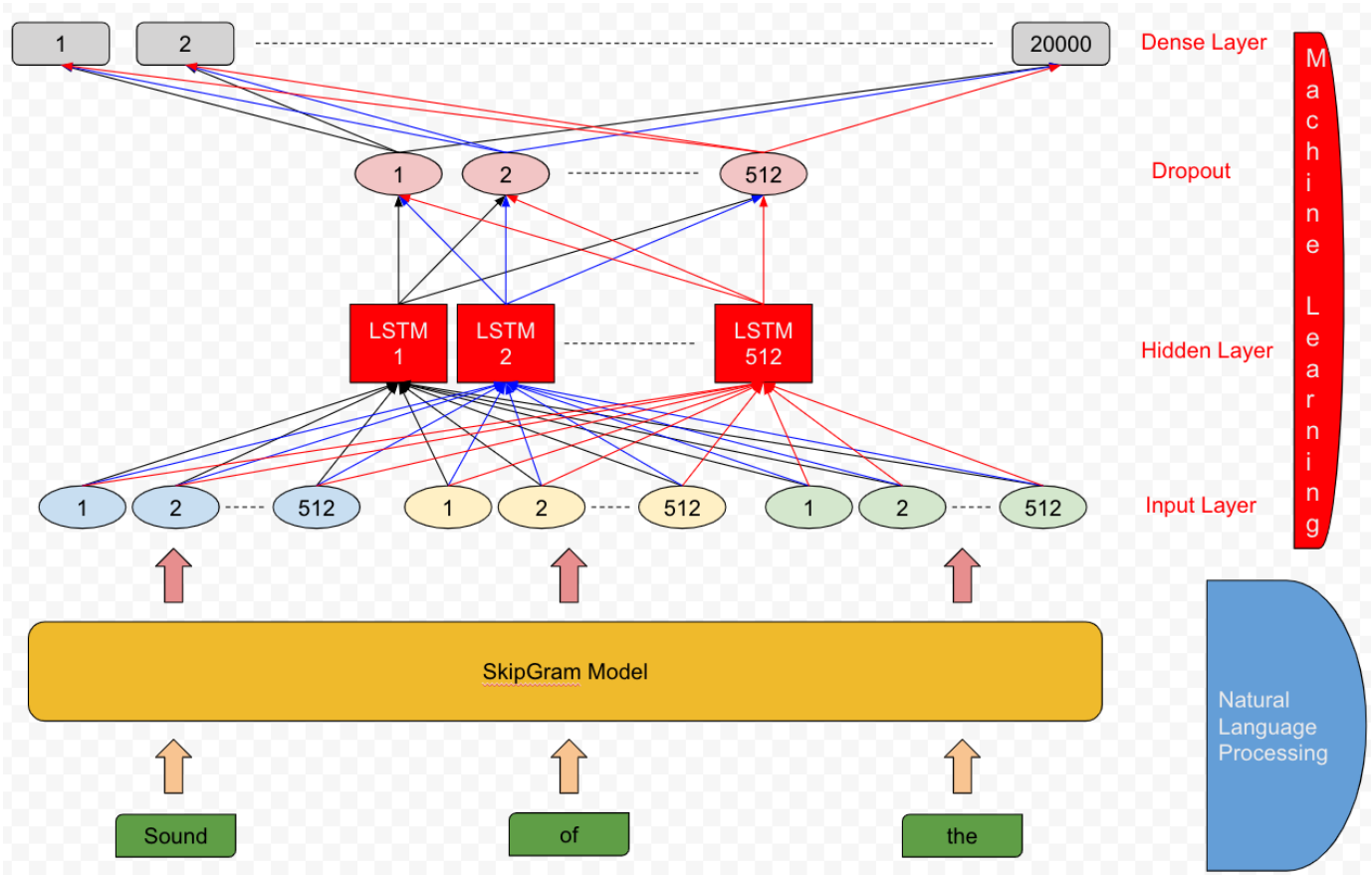
I figured out the problem and I found this solution: the solution was to vectorized our words, which mean that each word that we have should be converted to embedding vector of 512 length. I chose this length according to research belongs to Microsoft in the following link: https://arxiv.org/pdf/1804.08473.pdf

It's not enough just to assign each word with a unique random vector, we should use a Natural Language Processing model called SkipGram to convert our word to embedding vectors. https://arxiv.org/pdf/1301.3781.pdf

The following picture specifies the architecture of the skip-gram model that we're using:

Embedding vectors are not only useful just for an input to our RNN, but also to know the association between the words that we have and which word belong to which value.

After vectorizing our words we build the new RNN-LSTM model to take 3 embedding vectors according to our sequence length as input and to give us the next word as a dense vector as an output. The architecture is in the following figure:



The training process: