Generating Questions with Custom Data

Emre Çoban1 Alihan Sağöz1

Abstract

Question generation given a context passage is one of the fundamental profits of artificial intelligence in education. In this paper, we try to introduce a sequence-to-sequence deep learning model using LSTM. On the other hand, as you can see from our title, we also will add some data by hand to extend our data from SQuAD (Stanford Question Answering Dataset) [[1](#page3)]. We think this project could help teachers or anyone who wants questions from the given context. A variety of structures can be used to achieve this project. We ran into a variety of choices. Of course, LSTM and GRU were the first ones we ran into, and we decided to use LSTM. But since it is a progression report we think we might change some things with our model. However, we believe that no matter what we achieve with our project still It will be a great experience for us.

1. Introduction

Questions are important for understanding everything around us. We can use them to understand and know more about the environment. That is the main reason why we chose this project as our topic. We wanted to discover the opportunities that artificial intelligence can provide in education. We are mainly interested in understanding Natural Language Processing and understanding more about processing sequence data. Of course, our project is considered a sequence-to-sequence model. This means we will input a sequence that we consider as our context passage and answer and we will get an output of sequence which will be our generated question. To achieve this we at the first had a look at encoder-decoder models. Figure 1, for example, shows an example encoder-decoder structure.

We also encountered transformers architecture as an alternative to LSTM which seems more efficient since it uses self-attention and positional embeddings.

Diagram

Description automatically generated

Figure 1: Sample encoder-decoder structure [[2](#page3)].

The idea is to use fixed or learned weights that encode information related to a specific position of a token in a sentence. The first point is the main reason why transformers do not suffer from long dependency issues. The original transformers do not rely on past hidden states to capture dependencies with previous words. They instead process a sentence as a whole. That is why there is no risk to lose (or "forget") past information. Moreover, multi-head attention and positional embeddings both provide information about the relationship between different words.

But when we kept working on it for now, we decided to move on with LSTM (Long Short-Term Memory) [[3](#page3)] (figure 2) encoder-decoder model. Because we understood and studied it a little bit more and we think it is a good model to start with this project.

Diagram, schematic

Description automatically generated

Figure 2: LSTM cell structure [[4](#page3)].

LSTM is an artificial neural network used in the fields of artificial intelligence and deep learning. Unlike feedforward neural networks, LSTM has feedback connections. LSTM is suitable for tasks like machine translation which is a sequence-to-sequence task just like our project.

For those who don’t know why are RNN (Recurrent Neural Network) [5] not enough for this kind of topics? To answer this question to start with, RNN also has feedback connections, and it was suitable for tasks that are suitable for LSTM in the past. Main problem of RNN is that when the input sequence gets longer and longer it has this vanishing gradient problem, figure 3 shows visualization of this vanishing gradient problem.

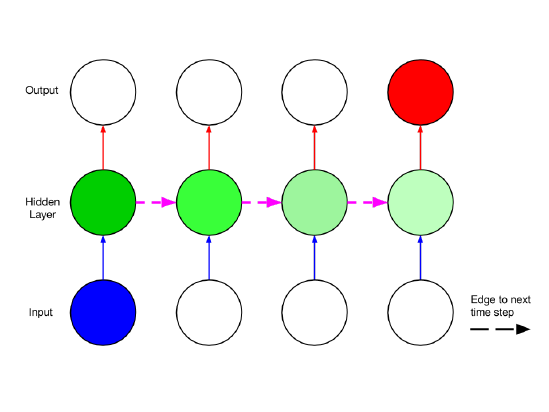


Figure 3: Visualization of Vanishing Gradient [[6](#page3)].

this problem causes model to not update weights and biases accordingly for the task. So, we go with LSTM because it deals with this problem using more gates.

LSTM has 3 important gates an input gate, an output age and a forget gate. The input gate as it is obvious from its name it takes input from a cell before or from input. The forget gate controls whether to forget or keep the information in the cell

To summarize we will move on with encoder-decoder model using LSTM because we understood and studied it more right now. But maybe we can change our idea and use Transformers or fine tune a pretrained model such as BERT (Bidirectional Encoder Representations from Transformers) [7].

2. Related Work

We found a lot of work related to our project. There are many studies including LSTM network for question answering [8] which is very similar research are to our project and there is also one study about question generation using transformers [9] ### TRANSFORMERS, Bİ-LSTM QUESTİON GENERATİON , QUESTİON ANSWERİNG 4-5 PAPER NELER FARKLI, HANGİ PAPERLARDAN NELER ALDIK VE KISA SÜREDE ANLADIĞIMIZ KISMI ALDIK VS .VS GİBİ ###

3. The Approach

3.1. Dataset The data we will use is hosted at drivendata.org [[1](#page3)] and is from the earthquake happened in Gorkha, Nepal. It consists of 260,601

buildingsand38featuressuch as age, height, area, land surface percentage, number of floors, etc. which are labeled as 1,2,3; corresponding to low-medium-high

In equation [1](#page2) the term (1 pt) is added to the standard cross entropy loss. By adding this term focal loss is decreas-ing the loss coming from the labels that are easily learned and makes the model focus on hard to learn labels.

After preparing the dataset, the K-NN classifier was applied, resulting inh 70.9% accuracy. Then, weighted K-NN re-

(2)

sulted with 71.33%. For weighted K-NN, euclidian distance is used as the distance metric. The reason for using K-NN is it is very simple to apply, especially in the beginning, to see if there is a pattern between features and labels; also it has few parameters to tune (number of neighbors, distance metric); and it doesn’t require the classes to be linearly separable.

levels of damage, respectively. There is class imbalance as it is visualized in Figure 1.

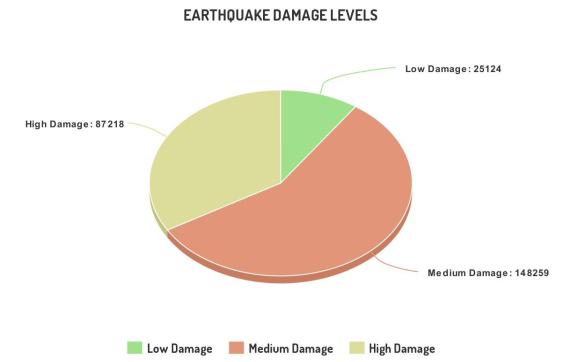


Figure 1: Distrubution of labels

Predicting damage may occur on a building just by knowing its features is quite hard since this is a very complex prob-lem that requires lots of parameters than just the building features. Since there is no known approach to solve this problem, we will first apply basic algorithms such as K-NN, Naive Bayes, Logistic Regression and Random Forest. We investigate if this problem is solvable by machine learning al-gorithms by examining how these algorithms behave. In our investigations, we observed a huge class imbalance in the data which can be seen in Figure 1. We have some prelimi-nary results with these basic algorithms which are discussed in the experimental results section. Our main approach will be to train a deep neural network model on this problem. However, we first need to think about class imbalance prob-lems to prevent overfitting. There are many approaches to solve the class imbalance problem while training a neural network such as focal loss and Sthochastic Gradient Descent with Warm Restart. These approaches are explained in the sections below.

3.2. Focal Loss

Focal Loss[[4](#page4)] is a loss function to reduce the problem of class imbalance with simple modifications to the Cross En-tropy Loss. Focal loss is defined as:

|  |  |
| --- | --- |
| F L(pt) = (1 pt) log(pt) | (1) |

Here if p is defined as the probability distribution on the classes calculated by the model :

(

3.3. Stochastic Gradient Descent with Warm Restarts

While training Deep Learning models, it is very common for the model to stuck in local optima. In this context, one of the suggested approaches to reach better optima in the parameters space is the use of restarting Stochastic Gradient Descent strategies. In Stochastic Gradient Descent with Warm Restarts -which is one of these strategies- suggested in [[5](#page4)], learning is being restarted with some values, quickly and periodically and decremented on schedule. These restart and value assignments for i’th run are done according to the equation below:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t = mini+ 2 | | maxi | mini |  | 1 + cos | Ti | |  | (3) |
|  | 1 |  |  |  |  |  | Tcur |  |  |

Here, Tcur corresponds to the number of epochs after the last restart, where Ti corresponds to the number of epochs of the current restart and at every restart, Tmul is incremented by the coefficent, represented with:

|  |  |
| --- | --- |
| Ti = Tcur + Ti Tmul | (4) |

4. Experimental Results

Dataset Description: The dataset contains damage levels of 260,000+ buildings with 38 features. Most of the features have discrete values, however there are 8 features which are categorical. First, we converted these features into discrete forms, e.g 3 distinct categories of a feature took values ”0, 1, 2”. After conversion, the dataset is splitted into Train (80%) and Test (20%) sets.

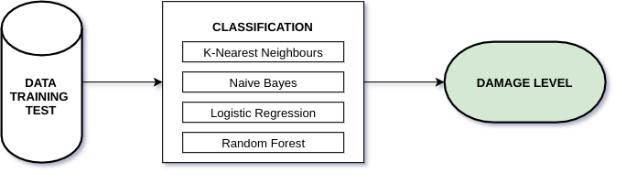


Figure 2: Flowchart of our experiments

4.1. K-nearest Neighbours Algorithm

pt =

p if y = 1

1 p else

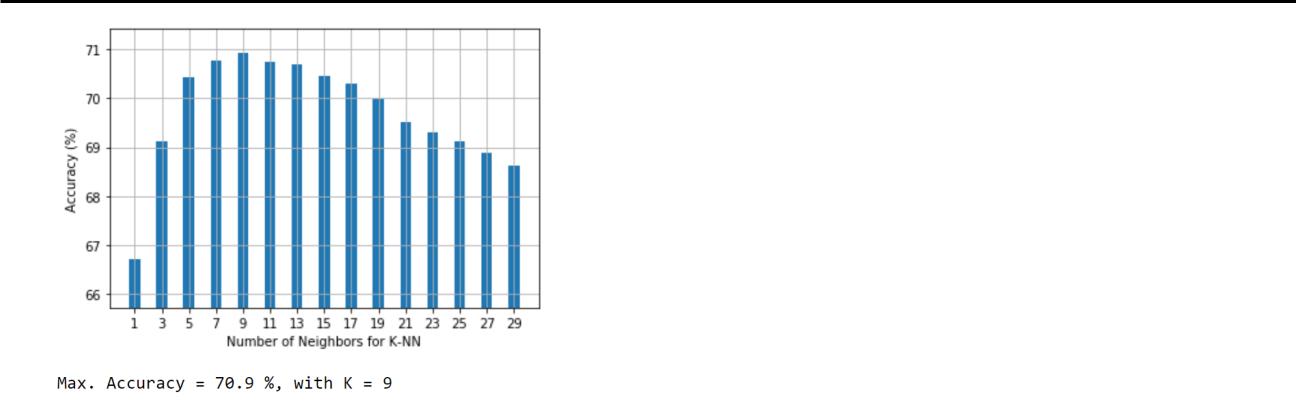


Figure 3: Accuracies for different number of neighbors

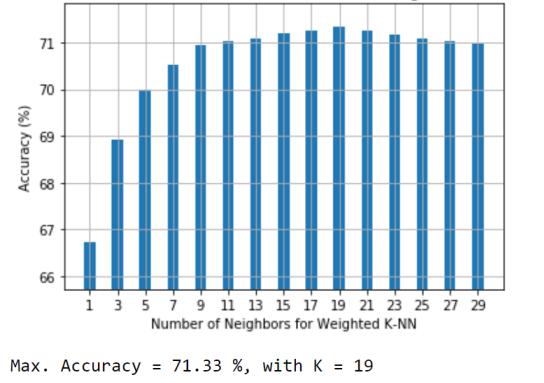


Figure 4: Accuracies for different number of neighbors, Weighted K-NN

4.2. Naive Bayes

We applied Gaussian Naive Bayes Classifier as our second approach. Each sample was taken as a distinct vector of length 38, and they were assumed conditionally independent and coming from a normal distribution. This approach was not successful and resulted in 46.7% accuracy. There are 2 major reasons for this outcome, one of these is some of the features are boolean and not coming from a normal distribution,

4.3. Logistic Regression

Our third approach was using Multinomial Logistic Re-gression (LR). The reason we choose LR is that it is highly interpretable, it doesn’t require features to be scaled, doesn’t need any tuning and its outputs are well-discriminated prob-abilities. However, the LR could not reach K-NN’s accuracy (57.3%). The main reason behind this is ”class imbalance”. When we look at the predictions of the model we observed that most of the predictions are made on medium level dam-age. This shows that class imbalance is very effective in the predictions of the

logistic regression model.

4.4. Random Forest

Random Forest is another machine learning method that operates by constructing multiple decision trees. The final decision is made based on the majority of the trees and is chosen by the random forest. The advantages of using a random forest algorithm are; it reduces the overfitting, gives higher accuracy than 1 decision tree since it reduces the variance by ensembling different decision trees, and runs efficiently on large datasets. In our experiments, random forest resulted in 71% accuracy with max depth = 28. We observed that the higher depth results in higher training performance, for random forest model. But it overfits and after a point increasing maximum depth does not increase the test set’s accuracy. The training procedure can be seen in Figure 5.

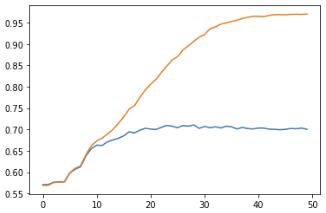


Figure 5: Training of a random forest model by adjusting max depth orange curve is for training set, blue curve is for test set.

|  |  |
| --- | --- |
| Algorithm | Accuracy (%) |
| K-NN | 70.9 |
| Weighted K-NN | 71.33 |
| Naive Bayes | 46.7 |
| Logistic Regression | 57.3 |
| Random Forest | 71.0 |

Table 1. Accuracies of different methods

References

[[4](#page3)]. https://www.researchgate.net/figure/Structure-of-the-LSTM-cell-and-equations-that-describe-the-gates-of-an-LSTM-cell\_fig5\_329362532

[[5](#page3)]. <https://en.wikipedia.org/wiki/Long_short-term_memory>

[6]

[7] <https://arxiv.org/abs/1810.04805>

[8] https://arxiv.org/abs/1707.01961

[9] https://arxiv.org/pdf/1909.05017.pdf

[10]