# **Natural Language Inference(NLI)**

Naralachetty Navitha, Siddharth Pullakhandam

Department of Computer Science, University of Wisconsin-Milwaukee

CS 723: Natural Language Processing

Susan McRoy, Ph.D.,

December 19, 2022

# **Abstract**

This is a task based on Natural Language Inference (NLI). We are given two sentences for this task, referred to as the premise and the hypothesis. If the "premise" is correct, then we must decide whether the "hypothesis" is true (entailment), untrue (contradiction), or indeterminate (neutral).We used the Stanford Natural Language Inference (SNLI) dataset for this experiment.

**Introduction**

Natural Language Inferencing (NLI) task is one of the most important subsets of Natural Language Processing (NLP) which has seen a series of development in recent years. There are standard benchmark publicly available datasets like [Stanford Natural Language Inference (SNLI) Corpus](https://nlp.stanford.edu/projects/snli/), [Multi-Genre NLI (MultiNLI) Corpus](https://cims.nyu.edu/~sbowman/multinli/), etc. which are dedicated to NLI tasks. Few state-of-the-art models trained on these datasets possess decent accuracy. Natural language inference studies whether a hypothesis can be inferred from a premise, where both are a text sequence. In other words, natural language inference determines the logical relationship between a pair of text sequences.

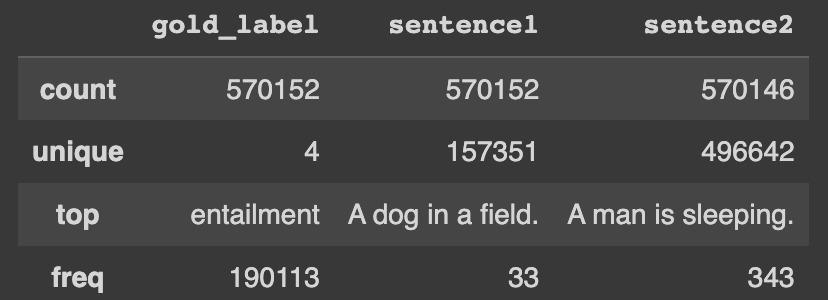
Such relationships usually fall into three types:

1. Entailment: the hypothesis can be inferred from the premise.
2. Contradiction: the negation of the hypothesis can be inferred from the premise.
3. Neutral: all the other cases.

# **MODELS**

# **LONG SHORT TERM MEMORY (LSTM)**

The LSTM models extend the RNNs’ memory in order to keep and learn long-term dependencies of inputs. The LSTM memory is called a “gated” cell, which is functioned to make the decision of preserving or ignoring the memory information. To be precise, an LSTM model consists of three gates, forget, input, and output gates. The forget gate makes the decision to preserve or remove the existing information. The input gate controls the extent to which the new information should be added into the memory. The output gate specifies what part of the LSTM memory contributes to the output.

Step 1:- We have to load the SNLI dataset.  
Step-2 :- Now import all the packages like numpy, pandas, keras.  
Step 3 :- Now we have to clean the data such that we can send it to the model.  
Step 4 :- The below image is the description of the whole data.

Step 5 :- We have to remove the Stop Words and also make all the letters into lower case.  
Step 6 :- Resizing the size of the dataset such that all the data points are having the same size and we can input this data to the model.   
Step 7 :- Now after we have all the data which is cleaned and also they have the same size now it’s ready to be inputted to the LSTM model.  
Step 8 :- After training the data we got plot the accuracy curve as shown below and the accuracy turns out be around 84%.  
Step 9 :- Now our model is ready for testing.

Chart, line chart

Description automatically generated

Figure 1.1 ACCURACY GRAPH OF LSTM

**BIDIRECTIONAL ENCODER REPRESENTATIONS**

**FROM**

**TRANSFORMER (BERT)**

BERT stands for Bidirectional Encoder Representations from Transformers. BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be finetuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task specific architecture modifications. One among those is a tool called BERT (Bidirectional Encoder Representations from Transformers), developed at Google. BERT is a deeply bidirectional, unsupervised language representation, pre-trained using only a plain text corpus. We can use BERT to obtain vector representations of documents/ texts. These vector representations can be used as predictive features in models.

Diagram

Description automatically generated

Figure 1.2 BERT MODEL ARCHITECTURE FOR TEXT PAIRS

**Working of BERT MODEL:**

**Data Pre-Processing:** We will begin by extracting the required data columns that are gold\_label, sentence1, and sentence2. Analyzing the Data: After loading the dataset into dataframes, we will analyze the data and check if we have labels for all the rows. Preparing the dataset: After analyzing, we will prepare our dataset to feed the pre-trained BERT model. Now we will load the vocabulary and token-index mapping from the transformers library using BertTokenizer. To ensure that every sentence is the same length, we will specify the maximum length for both sequences and individual sentences. We will define some functions to tokenize the sentences, to reduce the length of the sentences to max\_input\_length, to trim the sentence to max\_sentence\_length. After trimming the sentences, we will add the [CLS] token at the start of sentence1 and [SEP] token at the end of sentence1 and sentence2. After that, we will use the sequence to obtain the attention mask. The model benefits from the attention mask’s assistance in identifying the padding and tokens used during batch preparation. Model Training and Testing: In the beginning, we will download the pre-trained BERT-Base model from transformers. We will employ the BERT architecture with a single additional linear layer for output prediction. The output layer, which in this case has a size of three, will get the final hidden layer corresponding to the [CLS] token. Load the pre-trained BERT-Base model. Define a custom function to calculate the accuracy of our model. The evaluate function to evaluate the performance of our model. We got an accuracy of 85% on the training dataset and 89% accuracy on the testing dataset. A higher N\_EPOCH value will increase accuracy.

**PROJECT REPORT:**

This project includes a natural language inference (NLI) model, developed by fine-tuning Transformers on the SNLI corpus. Highlighted Features.

Given two sentences (premise and hypothesis), Natural Language Inference (NLI) is the task of deciding if the premise entails the hypothesis, if they are contradiction or if they are neutral.

1) Models based on BERT-(base, large).

2) Implemented using PyTorch (1.5.0).

3) Training the Bert-large model only requires around 6GB GPU memory (with batch size 8).

4) Easy interface: A user-friendly interface is provided to use the trained models.

5) All source code: All source code for training and testing the models is provided.

**Note:** That the code we have created has been primarily developed on Google Collab. Data was uploaded to google drive. So, we mount google drive and import datasets from drive to google collab VM.

**WORKING PROCESS :**

Step 1: At first, We need to import the required libraries such as numpy, pandas , matplotlib, pyplot, seaborn.

Step 2: Load the dataset of Stanford Natural Language Inference Corpus - train, dev, test in CSV format. Then we made sure that all the datasets have the same labels (neutral, contradiction and entailment.

Step 3: To visualize the proportion of gold labels in the form of a pie chart where it shows that both contradiction and entailment have both 33.3% and 33.2% sentences show that they are neutral.

Step 4: After that, we looked at the lengths of Sentences (train only) for both sentence 1 and sentence 2.

Step 5: We tried to visualize the distribution of the lengths of sentences 1 and 2. Then for sentences 1 and 2 we checked for the minimum and the maximum count of words.

Step 6: **Preprocessing:** Now, for the data preprocessing we omitted the rows having the gold Label, "-", null indexes and irrelevant columns.

Step 7: **Frequency Analysis:** To import more libraries we used the most important nltk library of Natural Language processing, import Counter from collections, s

topwords from the nltk corpus, word tokenizer from nltk tokenize, WordNetLemmatizer from nltk stem and ngrams from nltk. And to download the nltk.download('popular') to download all the collection and necessary dataset/models for specific functions to work. Tokenization of the english sentences , to remove all the stop words and Lemmatizing(become, becomes, becoming🡪become) and return all the new tokens. Connect all the sentences in the preprocessed training set.

Step 8: To Visualize the frequent words in the train dataset and the count the words we have plotted the graph.

Step 9: For Bert, At first, we will need to set up the environment and install some libraries. We have used torch.legacy. Therefore, we have installed transformers using pip since we will be using pre-trained BERT model from transformers.

install torchtext 0.10.0 using pip

pip install transformers~=2.11.0

Step 10: Imported the libraries torch,torch.utils.data imported TensorDataset, random\_split, DataLoader , RandomSampler, SequentialSampler. Then from transformers import BertTokenizer. Tokenized using the Bert Tokenizer because BERT utilizes WordPiece vocabulary, which has a vocabulary capacity of roughly 30,000. We will use the same tokenizer used to train the pre-trained BERT model. tokenize the from the pre-trained (**'bert-base-cased'**) and get the maximum number of words.

Step 11: Then, Word Embedding we will create a function in order to get the word ID and attention mask. An attention mask is a series of 1s the same length as input tokens.To train and evaluate to make the tensor dataset,load ,test ,evaluate and test the data loader.

Step 12: Now, from transformers import the BertForSequenceClassification, AdamW, BertConfig. And load the pre-trained Bert Model. To measure the computing time we imported the time, datetime and garbage collection to save memory. Later on the to Train and Evaluation, and we set the maximum epoch\_value to atleast 3.

Step 13: To check the Model Loss taking place we plotted a graph by importing the matplotlib.ticker to plot the train loss, Evaluation loss and Average loss.

Step 14: **Prediction and model Performance checking:** Now to predict ad model performance checking and extract he relevant information from prediction we predicted the label list.And also analyzed the performance score of the model by the producing the fields like precision, recall and accuracy.

Step 15**: Confusion Matrix :** To also visualize the 2d model of the prediction we have plotted the confusion matrix in order to understand the prediction accurately.

**Future Aspects**

* **Increased use in natural language generation:**

Natural language inference can also be used in natural language generation, where it can help generate more coherent and realistic sentences by understanding the relationship between different sentences in a text.

* **Integration with other natural language processing tasks:**

Natural language inference is likely to be integrated with other natural language processing tasks, such as named entity recognition and part-of-speech tagging, to create more comprehensive and effective natural language processing systems.