```
!pip install torchtext==0.10.0
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting torchtext==0.10.0
  Downloading torchtext-0.10.0-cp38-cp38-manylinux1_x86_64.whl (7.6 MB)
     |████████████████████████████████| 7.6 MB 29.3 MB/s
Requirement already satisfied: tqdm in /usr/local/lib/python3.8/dist-packages (from torchtext==0.10.0) (4.64.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (from torchtext==0.10.0) (1.21.6)
Collecting torch==1.9.0
  Downloading torch-1.9.0-cp38-cp38-manylinux1_x86_64.whl (831.4 MB)
     |████████████████████████████████| 831.4 MB 15 kB/s
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (from torchtext==0.10.0) (2.23.0)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.8/dist-packages (from torch==1.9.0->torchtext==0.10.0) (4.4.0)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests->torchtext==0.10.0) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests->torchtext==0.10.0) (2022.9.24)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests->torchtext==0.10.0) (1.24.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.8/dist-packages (from requests->torchtext==0.10.0) (3.0.4)
Installing collected packages: torch, torchtext
  Attempting uninstall: torch
    Found existing installation: torch 1.13.0+cu116
    Uninstalling torch-1.13.0+cu116:
      Successfully uninstalled torch-1.13.0+cu116
  Attempting uninstall: torchtext
    Found existing installation: torchtext 0.14.0
    Uninstalling torchtext-0.14.0:
      Successfully uninstalled torchtext-0.14.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following depen
torchvision 0.14.0+cu116 requires torch==1.13.0, but you have torch 1.9.0 which is incompatible.
torchaudio 0.13.0+cu116 requires torch==1.13.0, but you have torch 1.9.0 which is incompatible.
Successfully installed torch-1.9.0 torchtext-0.10.0
```

```
!pip install transformers
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting transformers
  Downloading transformers-4.25.1-py3-none-any.whl (5.8 MB)
     |████████████████████████████████| 5.8 MB 31.0 MB/s
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.8/dist-packages (from transformers) (21.3)
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (from transformers) (2.23.0)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.8/dist-packages (from transformers) (4.64.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.8/dist-packages (from transformers) (6.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.8/dist-packages (from transformers) (3.8.0)
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1
  Downloading tokenizers-0.13.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (7.6 MB)
     |████████████████████████████████| 7.6 MB 67.9 MB/s
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.8/dist-packages (from transformers) (2022.6.2)
Collecting huggingface-hub<1.0,>=0.10.0
  Downloading huggingface_hub-0.11.1-py3-none-any.whl (182 kB)
     |████████████████████████████████| 182 kB 66.2 MB/s
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.8/dist-packages (from transformers) (1.21.6)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.8/dist-packages (from huggingface-hub<1.0,>=0.10.0->transformers) (4.4.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.8/dist-packages (from packaging>=20.0->transformers) (3.0.9)
```

```
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.8/dist-packages (from requests->transformers) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests->transformers) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests->transformers) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests->transformers) (2022.9.24)
Installing collected packages: tokenizers, huggingface-hub, transformers
Successfully installed huggingface-hub-0.11.1 tokenizers-0.13.2 transformers-4.25.1
```

```
# downloading the dataset from the url
!wget https://nlp.stanford.edu/projects/snli/snli_1.0.zip
```

```
--2022-12-08 19:45:35--  https://nlp.stanford.edu/projects/snli/snli_1.0.zip
Resolving nlp.stanford.edu (nlp.stanford.edu)... 171.64.67.140
Connecting to nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 94550081 (90M) [application/zip]
Saving to: ‘snli_1.0.zip’

snli_1.0.zip        100%[===================>]  90.17M  15.4MB/s    in 9.2s

2022-12-08 19:45:45 (9.77 MB/s) - ‘snli_1.0.zip’ saved [94550081/94550081]
```

```
#unzip the files
!unzip snli_1.0.zip
```

```
Archive:  snli_1.0.zip
   creating: snli_1.0/
  inflating: snli_1.0/.DS_Store
   creating: __MACOSX/
   creating: __MACOSX/snli_1.0/
  inflating: __MACOSX/snli_1.0/._.DS_Store
 extracting: snli_1.0/Icon
  inflating: __MACOSX/snli_1.0/._Icon
  inflating: snli_1.0/README.txt
  inflating: __MACOSX/snli_1.0/._README.txt
  inflating: snli_1.0/snli_1.0_dev.jsonl
  inflating: snli_1.0/snli_1.0_dev.txt
  inflating: snli_1.0/snli_1.0_test.jsonl
  inflating: snli_1.0/snli_1.0_test.txt
  inflating: snli_1.0/snli_1.0_train.jsonl
  inflating: snli_1.0/snli_1.0_train.txt
  inflating: __MACOSX/._snli_1.0
```

```
!ls snli_1.0
```

```
Icon        snli_1.0_dev.jsonl  snli_1.0_test.jsonl  snli_1.0_train.jsonl
README.txt  snli_1.0_dev.txt    snli_1.0_test.txt    snli_1.0_train.txt
```

```
import pandas as pd
```

```
# importing the dataset into dataframes
df_train = pd.read_csv("snli_1.0/snli_1.0_train.txt", sep="\t")
df_dev = pd.read_csv("snli_1.0/snli_1.0_dev.txt", sep="\t")
df_test = pd.read_csv("snli_1.0/snli_1.0_test.txt", sep="\t")
```

```
print(df_train.shape)
print(df_dev.shape)
print(df_test.shape)
```

```
(550152, 14)
(10000, 14)
(10000, 14)
```

```
df_train.head()
```

	gold_label	sentence1_binary_parse	sentence2_binary_parse	sentence1_parse	sentence2_parse	sentence1	sentence2	captionID	pairID
0	neutral	(((A person) (on (a horse))) ((jump...	((A person) ((is ((training (his horse...	(ROOT (S (NP (NP (DT A) (NN person)) (PP (IN o...	(ROOT (S (NP (DT A) (NN person)) (VP (VBZ is) ...	A person on a horse jumps over a broken down a...	A person is training his horse for a competition.	3416050480.jpg#4	3416050480.jpg#4r1n
1	contradiction	(((A person) (on (a horse))) ((jump...	((A person) ((((is (at (a diner))))...	(ROOT (S (NP (NP (DT A) (NN person)) (PP (IN o...	(ROOT (S (NP (DT A) (NN person)) (VP (VBZ is) ...	A person on a horse jumps over a broken down a...	A person is at a diner, ordering an omelette.	3416050480.jpg#4	3416050480.jpg#4r1c
2	entailment	(((A person) (on (a horse))) ((jump...	((A person) ((((is outdoors) ,) (on ...	(ROOT (S (NP (NP (DT A) (NN person)) (PP (IN o...	(ROOT (S (NP (DT A) (NN person)) (VP (VBZ is) ...	A person on a horse jumps over a broken down a...	A person is outdoors, on a horse.	3416050480.jpg#4	3416050480.jpg#4r1e
3	neutral	(Children (((smiling and) waving) (at c...	(They (are (smiling (at (their parents) ...	(ROOT (NP (S (NP (NNP Children)) (VP (VBG smil...	(ROOT (S (NP (PRP They)) (VP (VBP are) (VP (VB...	Children smiling and waving at camera	They are smiling at their parents	2267923837.jpg#2	2267923837.jpg#2r1n
4	entailment	(Children (((smiling and) waving) (at c...	(There ((are children) present))	(ROOT (NP (S (NP (NNP Children)) (VP (VBG smil...	(ROOT (S (NP (EX There)) (VP (VBP are) (NP (NN...	Children smiling and waving at camera	There are children present	2267923837.jpg#2	2267923837.jpg#2r1e

```
df_test.head()
```

	gold_label	sentence1_binary_parse	sentence2_binary_parse	sentence1_parse	sentence2_parse	sentence1	sentence2	captionID	pairID
0	neutral	((This (church choir)) (((sings (to (...	((The church) ((has (cracks (in (the c...	(ROOT (S (NP (DT This) (NN church) (NN choir))...	(ROOT (S (NP (DT The) (NN church)) (VP (VBZ ha...	This church choir sings to the masses as they ...	The church has cracks in the ceiling.	2677109430.jpg#1	2677109430.jpg#1r1n
1	entailment	((This (church choir)) (((sings (to (...	((The church) ((is (filled (with song)...	(ROOT (S (NP (DT This) (NN church) (NN choir))...	(ROOT (S (NP (DT The) (NN church)) (VP (VBZ is...	This church choir sings to the masses as they ...	The church is filled with song.	2677109430.jpg#1	2677109430.jpg#1r1e
2	contradiction	((This (church choir)) (((sings (to (...	(((A choir) (singing (at (a (baseball ...	(ROOT (S (NP (DT This) (NN church) (NN choir))...	(ROOT (NP (NP (DT A) (NN choir)) (VP (VBG sing...	This church choir sings to the masses as they ...	A choir singing at a baseball game.	2677109430.jpg#1	2677109430.jpg#1r1c
3	neutral	(((A woman) (with (((((a (green hea...	((The woman) ((is young) .))	(ROOT (NP (NP (DT A) (NN woman)) (PP (IN with)...	(ROOT (S (NP (DT The) (NN woman)) (VP (VBZ is)...	A woman with a green headscarf, blue shirt and...	The woman is young.	6160193920.jpg#4	6160193920.jpg#4r1n
4	entailment	(((A woman) (with (((((a (green hea...	((The woman) ((is (very happy)) .))	(ROOT (NP (NP (DT A) (NN woman)) (PP (IN with)...	(ROOT (S (NP (DT The) (NN woman)) (VP (VBZ is)...	A woman with a green headscarf, blue shirt	The woman is very happy.	6160193920.jpg#4	6160193920.jpg#4r1e

```
df_dev.head()
```

	gold_label	sentence1_binary_parse	sentence2_binary_parse	sentence1_parse	sentence2_parse	sentence1	sentence2	captionID	pairID
0	neutral	((Two women) ((are (embracing (while (...	((The sisters) ((are ((hugging goodbye ...	(ROOT (S (NP (CD Two) (NNS women)) (VP (VBP ar...	(ROOT (S (NP (DT The) (NNS sisters)) (VP (VBP ...	Two women are embracing while holding to go pa...	The sisters are hugging goodbye while holding ...	4705552913.jpg#2	4705552913.jpg#2r1n
1	entailment	((Two women) ((are (embracing (while (...	((Two woman) ((are (holding packages))...	(ROOT (S (NP (CD Two) (NNS women)) (VP (VBP ar...	(ROOT (S (NP (CD Two) (NN woman)) (VP (VBP are...	Two women are embracing while holding to go pa...	Two woman are holding packages.	4705552913.jpg#2	4705552913.jpg#2r1e

```
# extracting the required columns form the dataset
df_train = df_train[['gold_label','sentence1','sentence2']]
df_dev = df_dev[['gold_label','sentence1','sentence2']]
df_test = df_test[['gold_label','sentence1','sentence2']]
```

```
df_train.head()
```

	gold_label	sentence1	sentence2
0	neutral	A person on a horse jumps over a broken down a...	A person is training his horse for a competition.
1	contradiction	A person on a horse jumps over a broken down a...	A person is at a diner, ordering an omelette.
2	entailment	A person on a horse jumps over a broken down a...	A person is outdoors, on a horse.
3	neutral	Children smiling and waving at camera	They are smiling at their parents
4	entailment	Children smiling and waving at camera	There are children present

```
df_dev.head()
```

	gold_label	sentence1	sentence2
0	neutral	Two women are embracing while holding to go pa...	The sisters are hugging goodbye while holding ...
1	entailment	Two women are embracing while holding to go pa...	Two woman are holding packages.
2	contradiction	Two women are embracing while holding to go pa...	The men are fighting outside a deli.
3	entailment	Two young children in blue jerseys, one with t...	Two kids in numbered jerseys wash their hands.
4	neutral	Two young children in blue jerseys, one with t...	Two kids at a ballgame wash their hands.

```
df_test.head()
```

	gold_label	sentence1	sentence2
0	neutral	This church choir sings to the masses as they ...	The church has cracks in the ceiling.
1	entailment	This church choir sings to the masses as they ...	The church is filled with song.
2	contradiction	This church choir sings to the masses as they ...	A choir singing at a baseball game.
3	neutral	A woman with a green headscarf, blue shirt and...	The woman is young.
4	entailment	A woman with a green headscarf, blue shirt and...	The woman is very happy.

```
# Analyzing the data
df_train.groupby('gold_label').count()
```

gold_label	sentence1	sentence2
-	785	785
contradiction	183187	183185
entailment	183416	183414
neutral	182764	182762

```
# removing the entries from all train, dev and test datasets with label '-'
df_train = df_train[df_train['gold_label'] != '-']
df_dev = df_dev[df_dev['gold_label'] != '-']
df_test = df_test[df_test['gold_label'] != '-']
# analyzing the data
print(df_train.groupby('gold_label').count())
print(df_dev.groupby('gold_label').count())
print(df_test.groupby('gold_label').count())
```

```
               sentence1  sentence2
gold_label
contradiction     183187     183185
entailment        183416     183414
neutral           182764     182762
               sentence1  sentence2
gold_label
contradiction       3278       3278
entailment          3329       3329
neutral             3235       3235
               sentence1  sentence2
gold_label
contradiction       3237       3237
entailment          3368       3368
neutral             3219       3219
```

```
# dropping the rows from the data with NaN values
df_train = df_train.dropna(subset = ['sentence2'])

df_train.groupby('gold_label').count()
```

gold_label	sentence1	sentence2
contradiction	183185	183185
entailment	183414	183414
neutral	182762	182762

Pre-Processing

Converting the dataset into the form required by the pre-trained BERT-Base Model.

```
import torch
from transformers import BertTokenizer
```

```
# using the same tokenizer used in pre-training
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
```

```
Downloading: 100%  232k/232k [00:00<00:00, 253kB/s]
Downloading: 100%  28.0/28.0 [00:00<00:00, 656B/s]
Downloading: 100%  570/570 [00:00<00:00, 19.3kB/s]
```

```
# using the tokens from BertTokenizer
sep_token = tokenizer.sep_token
cls_token = tokenizer.cls_token
pad_token = tokenizer.pad_token
unk_token = tokenizer.unk_token

print(cls_token, sep_token, pad_token, unk_token)

#using the token ids
sep_token_idx = tokenizer.sep_token_id
cls_token_idx = tokenizer.cls_token_id
pad_token_idx = tokenizer.pad_token_id
unk_token_idx = tokenizer.unk_token_id
```

```
print(sep_token_idx, cls_token_idx, pad_token_idx, unk_token_idx)
```

```
    [CLS] [SEP] [PAD] [UNK]
    102 101 0 100
```

```
# defining the maximum length of the input sentences
max_input_length = tokenizer.max_model_input_sizes['bert-base-uncased']
# defining the maximum length of each sentence
max_sentence_length = 128
```

```
# function to tokenize the sentences using BertTokenizer
def tokenize_sentences(sentence):
  tokens = tokenizer.tokenize(sentence)
  return tokens
```

```
# function to reduce the size of the sentence to the max_input_length
def reduce_sentence_length(sentence):
  tokens = sentence.strip().split(" ")
  tokens = tokens[:max_input_length]
  return tokens
```

```
# function to trim the sentence to the max_sentence_length
def trim_sentence(sentence):
  # splitting the sentence
  sentence = sentence.split()
  # check if the sentence has 128 or more tokens
  if len(sentence) >= 128:
    sentence = sentence[:max_sentence_length]
  return " ".join(sentence)
```

Token type ids help the model to know which token belongs to which sentence. For tokens of the first sentence in input, token type ids contain 0 and for second sentence tokens, it contains 1.

```
# function to get the token type id's of the sentence-01
def token_type_ids_sent_01(sentence):
  try:
    return [0] * len(sentence)
  except:
    return []
```

```
# function to get the token type id's of the sentence-02
def token_type_ids_sent_02(sentence):
  try:
    return [1] * len(sentence)
```

```
  except:
    return []
```

Attention mask helps the model to know the useful tokens and padding that is done during batch preparation. Attention mask is basically a sequence of 1's with the same length as input tokens.

```
# function to get the attention mask of the given sentence
def attention_mask_sentence(sentence):
  try:
    return [1] * len(sentence)
  except:
    return []
```

```
# function to combine the sequences from lists
def combine_sequence(sequence):
  return " ".join(sequence)

# function to combine the masks
def combine_mask(mask):
  mask = [str(m) for m in mask]
  return " ".join(mask)
```

```
# trimming the sentences upto the maximum length
df_train['sentence1'] = df_train['sentence1'].apply(trim_sentence)
df_dev['sentence1'] = df_dev['sentence1'].apply(trim_sentence)
df_test['sentence1'] = df_test['sentence1'].apply(trim_sentence)

df_train['sentence2'] = df_train['sentence2'].apply(trim_sentence)
df_dev['sentence2'] = df_dev['sentence2'].apply(trim_sentence)
df_test['sentence2'] = df_test['sentence2'].apply(trim_sentence)
```

```
# adding the [cls] and [sep] tokens
df_train['t_sentence1'] = cls_token + ' ' + df_train['sentence1'] + ' ' + sep_token + ' '
df_dev['t_sentence1'] = cls_token + ' ' + df_dev['sentence1'] + ' ' + sep_token + ' '
df_test['t_sentence1'] = cls_token + ' ' + df_test['sentence1'] + ' ' + sep_token + ' '

df_train['t_sentence2'] = df_train['sentence2'] + ' ' + sep_token
df_dev['t_sentence2'] = df_dev['sentence2'] + ' ' + sep_token
df_test['t_sentence2'] = df_test['sentence2'] + ' ' + sep_token
```

```
# applying the BertTokenizer to the newly generated sentences
df_train['b_sentence1'] = df_train['t_sentence1'].apply(tokenize_sentences)
df_dev['b_sentence1'] = df_dev['t_sentence1'].apply(tokenize_sentences)
df_test['b_sentence1'] = df_test['t_sentence1'].apply(tokenize_sentences)
```

```
df_train['b_sentence2'] = df_train['t_sentence2'].apply(tokenize_sentences)
df_dev['b_sentence2'] = df_train['t_sentence2'].apply(tokenize_sentences)
df_test['b_sentence2'] = df_test['t_sentence2'].apply(tokenize_sentences)
```

```
# getting the token type ids for the sentences
df_train['sentence1_token_type'] = df_train['b_sentence1'].apply(token_type_ids_sent_01)
df_dev['sentence1_token_type'] = df_dev['b_sentence1'].apply(token_type_ids_sent_01)
df_test['sentence1_token_type'] = df_test['b_sentence1'].apply(token_type_ids_sent_01)

df_train['sentence2_token_type'] = df_train['b_sentence2'].apply(token_type_ids_sent_02)
df_dev['sentence2_token_type'] = df_dev['b_sentence2'].apply(token_type_ids_sent_02)
df_test['sentence2_token_type'] = df_test['b_sentence2'].apply(token_type_ids_sent_02)
```

```
# obtain the seqence from the tokenized sentences
df_train['sequence'] = df_train['b_sentence1'] + df_train['b_sentence2']
df_dev['sequence'] = df_dev['b_sentence1'] + df_dev['b_sentence2']
df_test['sequence'] = df_test['b_sentence1'] + df_test['b_sentence2']
```

```
# generating attention mask
df_train['attention_mask'] = df_train['sequence'].apply(attention_mask_sentence)
df_dev['attention_mask'] = df_dev['sequence'].apply(attention_mask_sentence)
df_test['attention_mask'] = df_test['sequence'].apply(attention_mask_sentence)
```

```
# combining the token type of both sentences
df_train['token_type'] = df_train['sentence1_token_type'] + df_train['sentence2_token_type']
df_dev['token_type'] = df_dev['sentence1_token_type'] + df_train['sentence2_token_type']
df_test['token_type'] = df_test['sentence1_token_type'] + df_test['sentence2_token_type']
```

Dropping the rows with NaN Sequence

```
df_dev.shape
```

```
(9842, 12)
```

```
from collections.abc import Iterable
testing_sequence = df_dev['sequence'].to_list()
for i in testing_sequence:
  if not isinstance(i, Iterable):
    print(i)
```

```
nan
nan
nan
nan
nan
nan
```

```
nan
nan
nan
nan
nan
nan
```

```
df_dev = df_dev.dropna(subset = ['sequence'])
df_dev.shape
```

```
(9830, 12)
```

```
# Converting the inputs to sequential for torchtext Field
df_train['sequence'] = df_train['sequence'].apply(combine_sequence)
df_dev['sequence']  = df_dev['sequence'].apply(combine_sequence)
df_test['sequence'] = df_test['sequence'].apply(combine_sequence)
```

```
df_train['attention_mask'] = df_train['attention_mask'].apply(combine_mask)
df_dev['attention_mask'] = df_dev['attention_mask'].apply(combine_mask)
df_test['attention_mask'] = df_test['attention_mask'].apply(combine_mask)
```

```
df_train['token_type'] = df_train['token_type'].apply(combine_mask)
df_dev['token_type'] = df_dev['token_type'].apply(combine_mask)
df_test['token_type'] = df_test['token_type'].apply(combine_mask)
```

```
# extracting the required columns
df_train = df_train[['gold_label', 'sequence', 'attention_mask', 'token_type']]
df_dev = df_dev[['gold_label', 'sequence', 'attention_mask', 'token_type']]
df_test = df_test[['gold_label', 'sequence', 'attention_mask', 'token_type']]
```

```
# saving the data in the files
df_train.to_csv('snli_1.0/snli_1.0_train.csv', index=False)
df_dev.to_csv('snli_1.0/snli_1.0_dev.csv', index=False)
df_test.to_csv('snli_1.0/snli_1.0_test.csv', index=False)
```

```
!ls snli_1.0
```

```
df_train.head()
```

```
# function to convert the attention_mask and token_type ids to int
def convert_to_int(ids):
  ids = [int(d) for d in ids]
  return ids
```

Create PyTorch Tensor using torchtext field

```
# importing the saved data from csv file
df_train = pd.read_csv('snli_1.0/snli_1.0_train.csv')
df_dev = pd.read_csv('snli_1.0/snli_1.0_dev.csv')
df_test = pd.read_csv('snli_1.0/snli_1.0_test.csv')
```

```
from torchtext.legacy import data
```

```
# text field for sequence
TEXT = data.Field(batch_first = True,
                  use_vocab = False,
                  tokenize = reduce_sentence_length,
                  preprocessing = tokenizer.convert_tokens_to_ids,
                  pad_token = pad_token_idx,
                  unk_token = unk_token_idx)
# label field for label
LABEL = data.LabelField()
# text field for attention mask
ATTENTION = data.Field(batch_first = True,
                       use_vocab = False,
                       tokenize = reduce_sentence_length,
                       preprocessing = convert_to_int,
                       pad_token = pad_token_idx)
# text field for token type ids
TTYPE = data.Field(batch_first = True,
                  use_vocab = False,
                  tokenize = reduce_sentence_length,
                  preprocessing = convert_to_int,
                  pad_token = 1)
```

```
fields = [('label', LABEL), ('sequence', TEXT), ('attention_mask', ATTENTION), ('token_type', TTYPE)]
```

```
train_data, valid_data, test_data = data.TabularDataset.splits(
                                        path = 'snli_1.0',
                                        train = 'snli_1.0_train.csv',
                                        validation = 'snli_1.0_dev.csv',
                                        test = 'snli_1.0_test.csv',
                                        format = 'csv',
                                        fields = fields,
                                        skip_header = True)
train_data_len = len(train_data)
```

```
# building the vocabulary for the labels
LABEL.build_vocab(train_data)
```

```
# using bucketiterator for preparing batches for training
BATCH_SIZE = 16
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
train_iterator, valid_iterator, test_iterator = data.BucketIterator.splits(
    (train_data, valid_data, test_data),
    batch_size = BATCH_SIZE,
    sort_key = lambda x: len(x.sequence),
    sort_within_batch = False,
    device = device)
```

Model Training

Using the pre-trained Bert_Model

```
from transformers import BertModel
bert_model = BertModel.from_pretrained('bert-base-uncased')
```

Using BERT architecture along with one linear layer for the output prediction

```
import torch.nn as nn
class BERTNLIModel(nn.Module):
    def __init__(self, bert_model, hidden_dim, output_dim,):
        super().__init__()
        self.bert = bert_model
        embedding_dim = bert_model.config.to_dict()['hidden_size']
        self.out = nn.Linear(embedding_dim, output_dim)

    def forward(self, sequence, attn_mask, token_type):
        embedded = self.bert(input_ids = sequence, attention_mask = attn_mask, token_type_ids = token_type)[1]
        output = self.out(embedded)
        return output
```

```
# loading the model
HIDDEN_DIM = 512
OUTPUT_DIM = len(LABEL.vocab)
model = BERTNLIModel(bert_model, HIDDEN_DIM, OUTPUT_DIM,).to(device)
```

```
# function to count the parameters of the model
def count_parameters(model):
  return sum(p.numel() for p in model.parameters() if p. requires_grad)

print(f'The model has {count_parameters(model):,} trainable parameters')
```

Using the Apex Nvidia a PyTorch extension for mixed precision and distributed training

```
%%writefile setup.sh

git clone https://github.com/NVIDIA/apex
cd apex
pip install -v --disable-pip-version-check --no-cache-dir ./
```

```
!sh setup.sh
```

Defining the loss function and optimizer for our model

```
from transformers.optimization import *
from apex import amp
import torch.optim as optim
optimizer = AdamW(model.parameters(),lr=2e-5,eps=1e-6,correct_bias=False)
def get_scheduler(optimizer, warmup_steps):
    scheduler = get_constant_schedule_with_warmup(optimizer, num_warmup_steps=warmup_steps)
    return scheduler
```

```
# using the cross entropy loss
criterion = nn.CrossEntropyLoss().to(device)
```

```
fp16 = True
if fp16:
    try:
        from apex import amp
    except ImportError:
        raise ImportError("Please install apex from https://www.github.com/nvidia/apex to use fp16 training.")
    model, optimizer = amp.initialize(model, optimizer, opt_level='O1')
```

```
# function to calculate the accuracy of model
def accuracy(pred, y):
    max_preds = pred.argmax(dim = 1, keepdim = True)
    correct = (max_preds.squeeze(1)==y).float()
    return correct.sum() / len(y)
```

```
max_grad_norm = 1

def train(model, iterator, optimizer, criterion, scheduler):
    epoch_loss = 0
    epoch_acc = 0
    model.train()
    for batch in iterator:
        optimizer.zero_grad() # clear gradients first
        torch.cuda.empty_cache() # releases all unoccupied cached memory
        sequence = batch.sequence
        attn_mask = batch.attention_mask
        token_type = batch.token_type
        label = batch.label
        predictions = model(sequence, attn_mask, token_type)
        loss = criterion(predictions, label)
        acc = accuracy(predictions, label)
        if fp16:
            with amp.scale_loss(loss, optimizer) as scaled_loss:
                scaled_loss.backward()
            torch.nn.utils.clip_grad_norm_(amp.master_params(optimizer), max_grad_norm)
        else:
            loss.backward()
        optimizer.step()
        scheduler.step()
        epoch_loss += loss.item()
        epoch_acc += acc.item()
    return epoch_loss / len(iterator), epoch_acc / len(iterator)
```

Model Testing

```
def evaluate(model, iterator, criterion):
    #print(iterator)
    epoch_loss = 0
    epoch_acc = 0
    model.eval()
    with torch.no_grad():
        for batch in iterator:
            sequence = batch.sequence
            attn_mask = batch.attention_mask
            token_type = batch.token_type
            labels = batch.label
            predictions = model(sequence, attn_mask, token_type)
            loss = criterion(predictions, labels)
            acc = accuracy(predictions, labels)
            epoch_loss += loss.item()
            epoch_acc += acc.item()
    return epoch_loss / len(iterator), epoch_acc / len(iterator)
```

```
import math
N_EPOCHS = 1

warmup_percent = 0.2
total_steps = math.ceil(N_EPOCHS * train_data_len * 1./BATCH_SIZE)
warmup_steps = int(total_steps*warmup_percent)
scheduler = get_scheduler(optimizer, warmup_steps)

best_valid_loss = float('inf')

for epoch in range(N_EPOCHS):

    train_loss, train_acc = train(model, train_iterator, optimizer, criterion, scheduler)
    valid_loss, valid_acc = evaluate(model, valid_iterator, criterion)

    if valid_loss < best_valid_loss:
        best_valid_loss = valid_loss
        torch.save(model.state_dict(), 'bert-nli.pt')

    print(f'\tTrain Loss: {train_loss:.3f} | Train Acc: {train_acc*100:.2f}%')
    print(f'\t Val. Loss: {valid_loss:.3f} |  Val. Acc: {valid_acc*100:.2f}%')
```

```
model.load_state_dict(torch.load('bert-nli.pt'))

test_loss, test_acc = evaluate(model, test_iterator, criterion)

print(f'Test Loss: {test_loss:.3f} |  Test Acc: {test_acc*100:.2f}%')
```

```
# function to get the results on custom inputs
def predict_inference(premise, hypothesis, model, device):
```

```
    # appending the 'cls' and 'sep' tokens
    premise = cls_token + ' ' + premise + ' ' + sep_token
    hypothesis = hypothesis + ' ' + sep_token

    # tokenize the premise and hypothesis using bert tokenizer
    tokenize_premise = tokenize_sentences(premise)
    tokenize_hypothesis = tokenize_sentences(hypothesis)

    # generate the token type ids of both premise and hypothesis
    premise_token_type = token_type_ids_sent_01(tokenize_premise)
    hypothesis_token_type = token_type_ids_sent_02(tokenize_hypothesis)

    # combining the tokenized premise and hypothesis to generate the sequence
    indexes = tokenize_premise + tokenize_hypothesis

    # converting the sequence of tokens into token ids
    indexes = tokenizer.convert_tokens_to_ids(indexes)

    # combining the premise and hypothesis tokens ids
    indexes_type = premise_token_type + hypothesis_token_type

    # generating the attention mask of the ids
    attention_mask = token_type_ids_sent_02(indexes)

    # creating the pytorch tensors of indexes, indexes_type, attention_mask
    indexes = torch.LongTensor(indexes).unsqueeze(0).to(device)
    indexes_type = torch.LongTensor(indexes_type).unsqueeze(0).to(device)
    attention_mask = torch.LongTensor(attention_mask).unsqueeze(0).to(device)

    # predicting to get the judgements
    prediction = model(indexes, attention_mask, indexes_type)

    prediction = prediction.argmax(dim=-1).item()

    return LABEL.vocab.itos[prediction]
```

```
premise = 'A black race car starts up in front of a crowd of people.'
hypothesis = 'A man is driving down a lonely road.'

predict_inference(premise, hypothesis, model, device)
```

```
premise = 'A soccer game with multiple males playing.'
hypothesis = 'Some men are playing a sport.'

predict_inference(premise, hypothesis, model, device)
```

```
premise = 'A smiling costumed woman is holding an umbrella.'
```