1 Problem Statement

In this project, I am attempting to reproduce the results obtained from this paper. In it, the authors implement a series of NLI models on the MNLI dataset. Then, they implement the same models on HANS, an NLI evaluation set that tests specific hypotheses about invalid heuristics that NLI models are likely to learn.[1]

As a reminder, Natural language inference (NLI) is the task of determining whether a hypothesis is true (entailment), false (contradiction), or undetermined (neutral) given a premise. For instance, the following sequence is tagged as *entailment*.[2]

(1) *Premise: A soccer game with multiple males playing.*
Hypothesis: Some men are playing a sport.

2 Data Preprocessing

I am working with the MNLI and HANS data sets. MNLI contains over 430000 sentence pairs, while HANS contains 60000 sentence pairs, 2000 belonging to each heuristic defined in the paper.[3] Each data set contains the following:

- `pairID`: ID for each premise/hypothesis pair
- `sentence1`: the premise sentence
- `sentence2`: the hypothesis sentence
- `sentence1_binary_parse`: a parse tree for the premise
  e.g. ((The authors) ((recognized (the doctor)) .))
- `sentence1_binary_parse`: a parse tree for the hypothesis
- `sentence1_parse`: a parse tree for the premise (with syntactical properties)
  e.g. (ROOT (S (NP (DT The) (NNS authors)) (VP (VBD recognized) (NP (DT the) (NN doctor))) (. .)))
- `sentence1_parse`: a parse tree for the hypothesis (with syntactical properties)
- `gold_label`: the relationship between the two sentences. In MNLI, it is one of *entailment*, *contradiction*, or *neutral*, whereas is it is one of *entailment* or *non-entailment* in HANS.

The HANS dataset also contains a `heuristic` feature which represents which heuristic the example belongs to, as well as a `subcase` feature to separate each heuristic into subcases. (You can find these subcases here (p.13))

Note that only `sentence1`, `sentence2`, and `gold_label` are used as features when implementing the model.

As for pre-processing, BERT tokenizes each sentence, and vectorizes them based on the longest sequence in the data set. Each is sentence is then padded as needed and input into an encoder. I will follow a similar pre-processing method to implement a RNN.

Additionally, to re-train BERT on a data set containing examples from HANS, I had to concatenate the MNLI dataset with a portion of the HANS dataset. Though, as mentioned above, both datasets are labelled differently. As a result, when training the second BERT model, I had to change all *contradiction* and *neutral* labels to *non-entailment*. This differs from the way I trained and tested the first BERT model, where I transformed *contradiction* and *neutral* to *non-entailment* after training and testing. Though, according to the paper, this should not have an effect on test results.

[1] https://arxiv.org/pdf/1902.01007.pdf
[2] http://nlpprogress.com/english/natural_language_inference.html
[3] Ibid

3 Machine Learning Model

I am using BERT for the first part of this project. More precisely, I am using BERT-Base uncased. BERT is a model built on the Transformer architecture[4], and pre-trained on a large corpus of unlabelled text (all of Wikipedia). Because of BERT's size, as well as the size of MNLI, fine-tuning was done using a TPU on Google Cloud Platform. Using GCP proved to be challenging at first since I had never used a cloud platform before.

According this article[5], what sets BERT apart from previous NLP models is its application of bidirectional training. That is, instead of reading test sequentially (from left to right), BERT reads the entire sequence of words at once. The paper introducing the model claims that a model that is bidirectionally trained can have deeper sense of language context. Though, the HANS dataset shows otherwise.

4 Preliminary Results

The evaluation results of BERT-Base uncased on the MNLI dataset were the following:

- `eval_accuracy = 0.8402`
- `eval_loss = 0.8079`
- `global_step = 36815`
- `loss = 0.7851`

This is in-line with what is out there for BERT-Base (BERT-Large can perform slightly better). In regard to the HANS dataset, results are shown below. Note that these results are those of a BERT model fine-tuned on the MNLI dataset only. My next task is to train BERT on examples from both MNLI and HANS.

Overall, BERT obtained 47.15% accuracy on HANS. This is almost 40% worse than on MNLI. We can separate these results into two classes: heuristic entailed results, which corresponds to examples in which the correct label is *entailed*, and heuristic non-entailed results, which corresponds to examples in which the correct label is *non-entailed*.

heuristic	entailed	non-entailed
lexical overlap	13.08%	74%
subsequence	1.62%	95.26%
constituent	0.42%	98.52%

Table 1: BERT accuracy on HANS dataset

From table 1, we see that BERT performed poorly on examples in which the correct label was *entailed*. Oddly, this is almost the opposite of the results obtained in the paper I'm attempting to reproduce. I have yet to find an explanation for this inconsistency. I am fairly confident it is not a label issue as, for examples at the beginning of the dataset, both results (mine and theirs) are similar. Most inconsistency occurs near the end of the dataset, perhaps suggesting that our results differ only in certain sub cases. Another reason could be that BERT-Base has been updated since then. I will need to continue to look for a explanation as to why this is occurring.

5 Next Steps

As mentioned above, I need to find an explanation for the differing results. To do so, I will begin by looking at differences between mine and the paper's results across heuristic sub cases.

Next, I will train my BERT model on a dataset that contains examples from both MNLI and HANS and compare the results. I have already built this dataset. Thus, all that is left to do is test the model on it. Also, this process might lead to an explanation to the inconsistency mentioned above.

After that, I will begin implementing a RNN to compare with BERT's results. It seems as though the best RNN for sequence classification is a bi-LSTM, so that will be the model I will use. Like BERT, the first test run on HANS will be of a model trained solely on MNLI. Then, I will re-train the model with examples from HANS

[4] https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html
[5] https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

and test it once more. This will be done using tensorflow.