TLF Capstone Project - Week 1 and 2

Team Members:
Bhavika Jain
Joseph Thachil
Sachin Lokesh

Agenda

Project Overview

Project: ML-based modeling of communication and decision making in design teams

Objective:

(i) Analyzing the communication patterns using natural language processing, and

(ii) Extending the computational model developed by Chaudhari et al. to include the content of communication during the design process, and

(iii) Running analysis on the dialogue exchange to draw insights on backup behavior of team members, task initiation,team social regulation etc.

(iv) ASME Design Conference: Research Poster Presentation, Presentation-only abstract, Student Hackathon, Design Essay Competition

ASME Design Conference Events/ Deadlines

Research Poster Presentation	20th April, 2021
Presentation-only Abstract	20th April 2021
Design Essay Competition	June 1st, 2021
Student Hackathon	July 30th, 2021

Progress before the capstone began

01

Read the following papers to understand the background of the problem statement:

- The Co Evolution of Team Communication and Design Performance in Engineering Design Teams - An Empirical Analysis
- Evaluating the Representativeness of Model Worlds

02

Analysed and tried to understand the following dataset files:

- chat_messages_cleaned.csv
- engine_design_cleaned.csv
- timespent_cleaned.csv

03

Performed **data cleaning** on the “body” column of chat_messages_cleaned.csv dataset to understand the nuances of text messages:

- Spelling correction | Punctuation removal | Lower casing | Stopwords removal
- Isolating frequent and rare words | Lemmatization to bring out the root form
- Chat words conversion to complete meaningful words

Week 1 Focussing Objective

- Tried to model the problem at hand as supervised intent classification
  - Used BERT model on 300 some labeled examples
  - Poor Performance with Training Accuracy = 71.55% and Test Accuracy = 53.8%

- 14 Intent Labels were used, such as:
  - Asking parameter value
  - Suggesting parameter value
  - Acknowledgement
  - Understanding dependency
  - Volunteering own global objectives information
  - Suggesting more local iterations
  - Requesting a broadcast etc.

Week 2 Focussing Objective

1. Unsupervised Intent Classification on text chat messages
   - Modelled the problem as semi-supervised intent classification using SOTA model - GANBERT
   - Compared the results given out by BERT and GANBERT on the parameters of accuracy, f1-score, precision, recall and overall loss

2. Research Poster Abstract

Updates

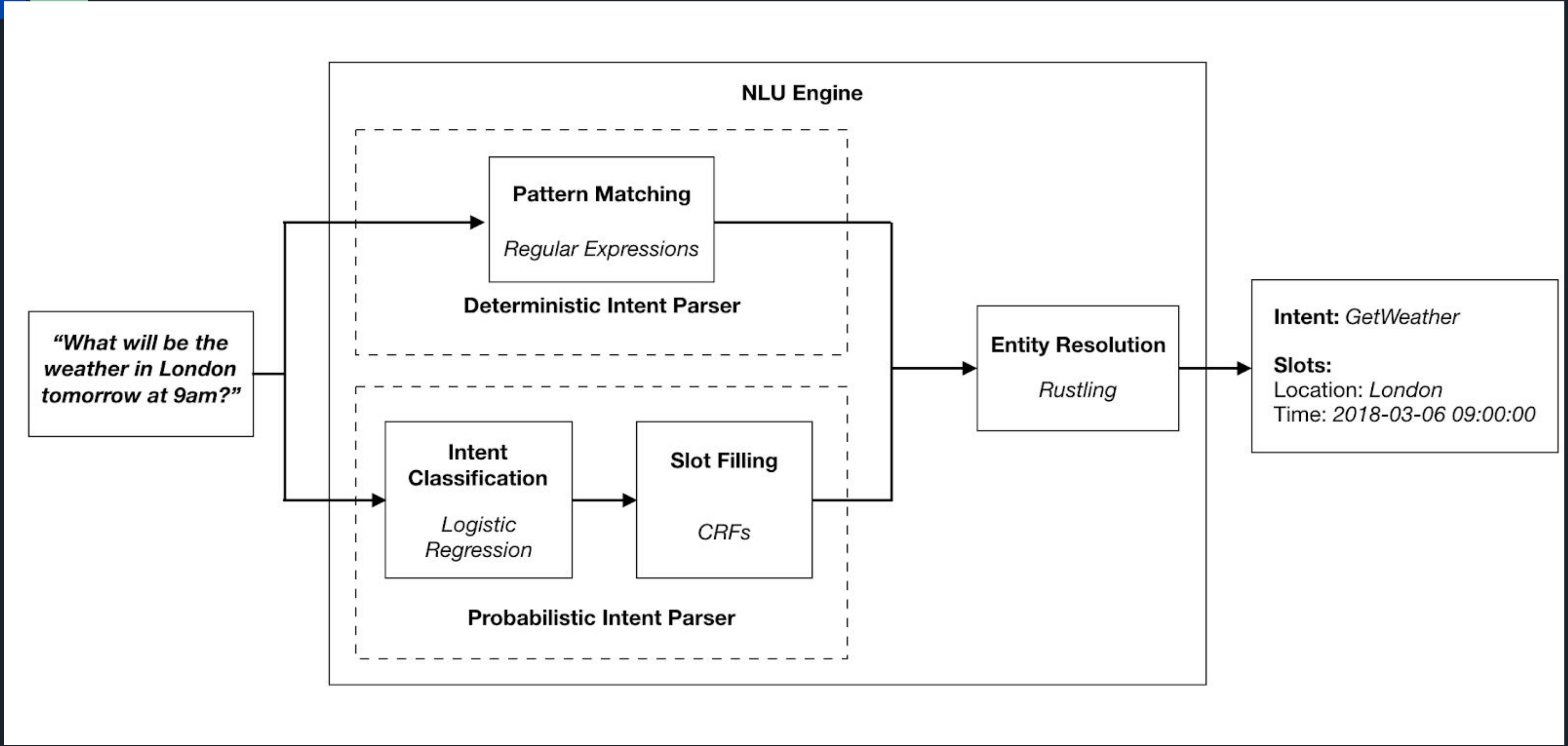
Amazon Mechanical Turk - Not Accessible in India

Alternative - Google Cloud Platform Labeling Task is in Progress

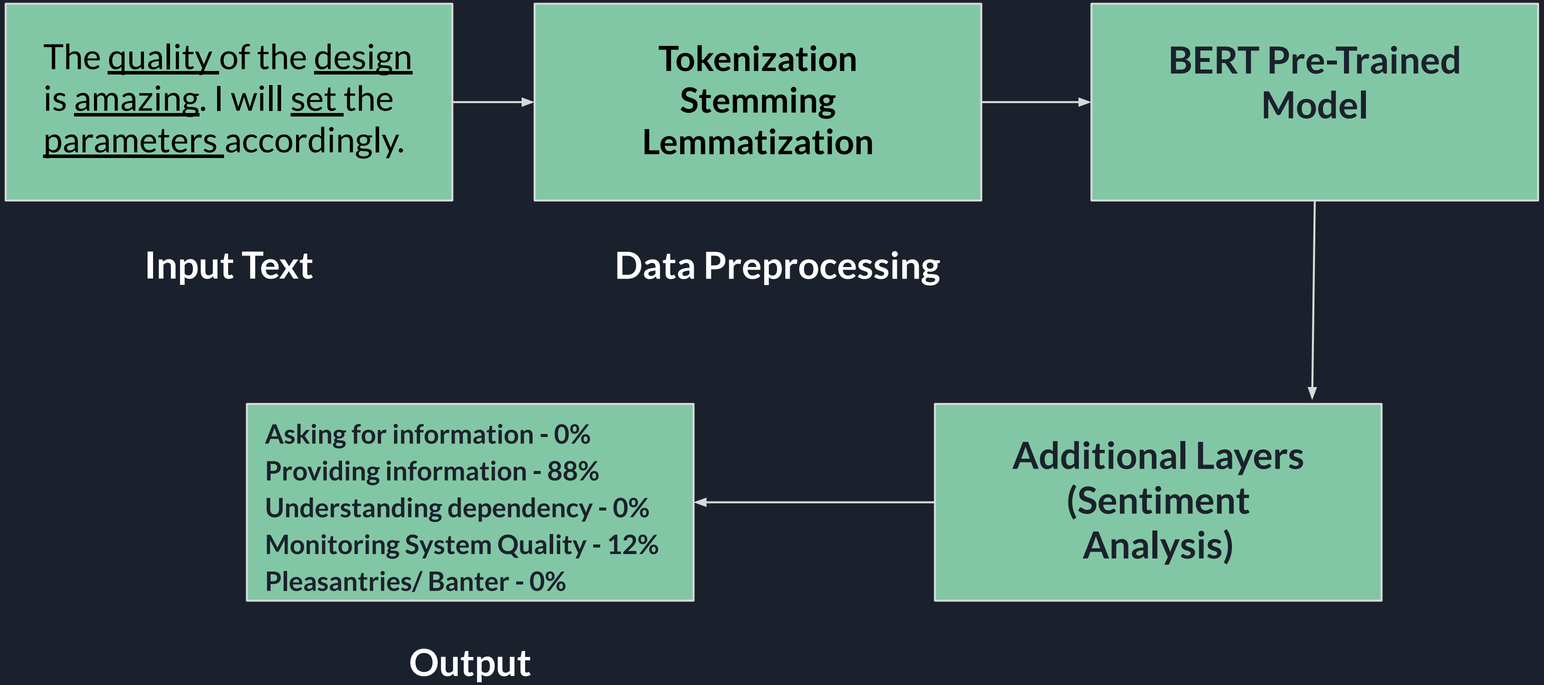
Final identified labels are -

- Asking for information
- Providing information
- Understanding dependency
- Monitoring System Quality
- Pleasantries/ Banter

NLU Pipeline



Work Plan Architecture


The quality of the design is amazing. I will set the parameters accordingly.
Input Text
Tokenization
Stemming
Lemmatization
Data Preprocessing
BERT Pre-Trained Model
Additional Layers (Sentiment Analysis)
Asking for information - 0%
Providing information - 88%
Understanding dependency - 0%
Monitoring System Quality - 12%
Pleasantries/ Banter - 0%
Output

What is BERT?

BERT is short for Bidirectional Encoder Representations from Transformers.

Pre-trained language models like BERT play an important role in many natural language processing tasks, such as:

- **Question Answering,**
- **Named Entity Recognition,**
- **Natural Language Inference,**
- **Text Classification etc.**

Why BERT?
The model works highly in parallel, so training speed is also extremely fast while improving machine translation performance.

What’s wrong with previous models?

- **Traditional machine translation is basically based on the Seq2Seq model.**

- **The model is divided into the encoder layer and the decoder layer, and is composed of RNN or RNN variants (LSTM, GRU, etc.).**

- **The encoder vector is the final hidden state produced from the encoder part of the model. This vector aims to encapsulate the information for all input elements in order to help the decoder make accurate predictions. It acts as the initial hidden state of the decoder part of the model**

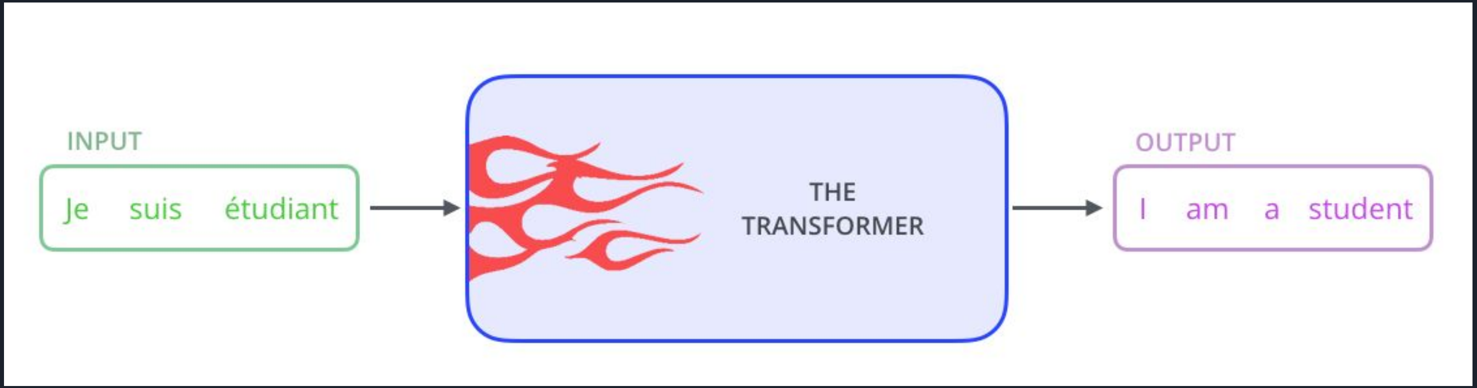
- **The main bottleneck of the Seq2Seq model is the need to compress the entire contents of the source sequence into this fixed-size vector.**

- **If the text is slightly longer, it is easy to lose some information of the text.**

How does it work?

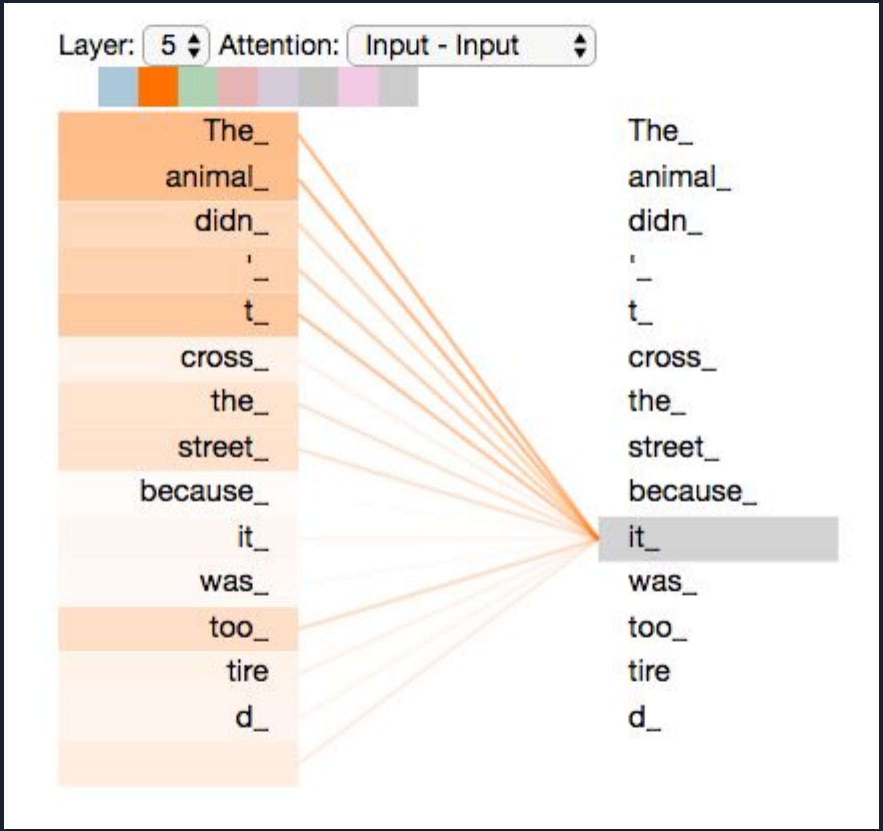
Attention

- The attention mechanism allows the decoder to look back at the source sequence hidden state , and then provide its weighted average as an additional input to the decoder.



Example: What does “it” in this sentence refer to?
Is it referring to the street or to the animal?

It’s a simple question to a human, but not as simple to an algorithm.



Prospects for next week's Discussion

- **Intent mining using past conversations using DB-SCAN clustering algorithm.**
- **Fine-tune the existing models at hand**

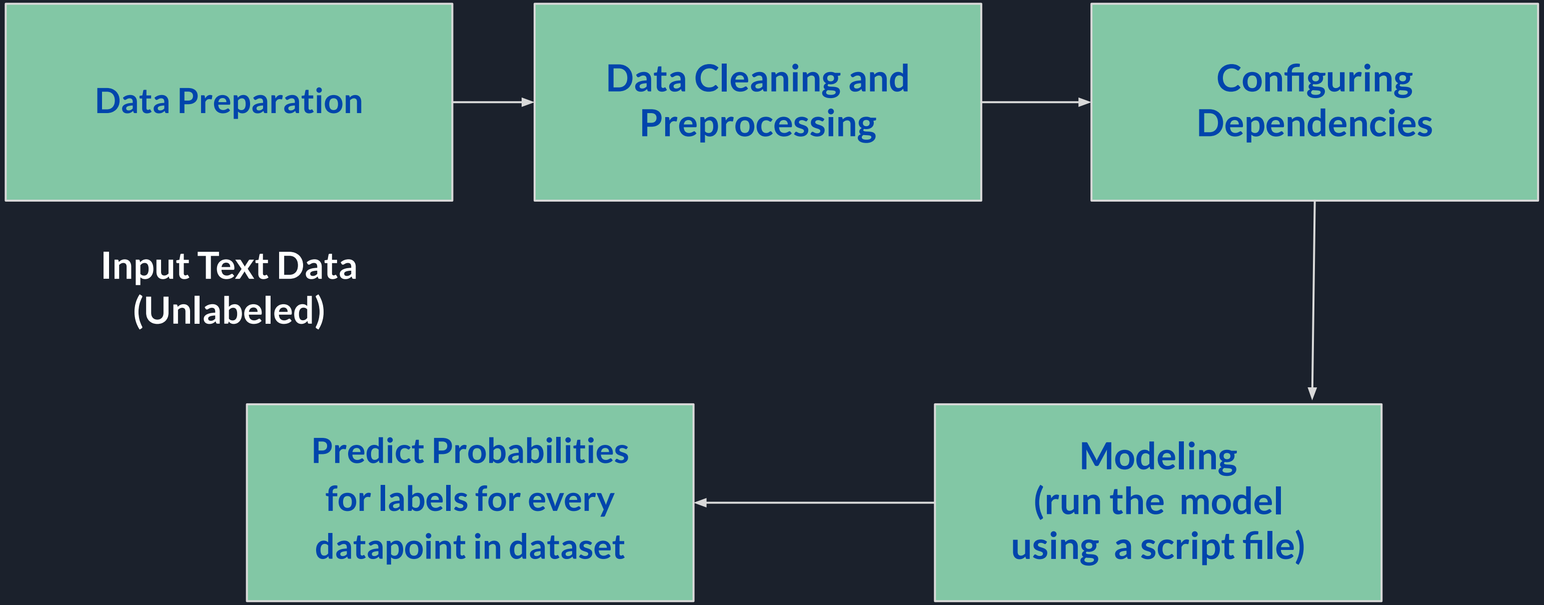
Discussion / Requests

1. A reference letter from the advisor for the research poster abstract

1. Final proofreading of the poster abstract

1. Official credentials/ access to journals, research papers and books

Work Plan Architecture


Data Preparation
Data Cleaning and Preprocessing
Configuring Dependencies
Modeling (run the model using a script file)
Predict Probabilities for labels for every datapoint in dataset

Input Text Data (Unlabeled)

Output Labels for all data points

Model Architecture - Semi-Supervised Intent Classification

Steps followed:

1. Annotated the data with labels (Initially decided 8 labels)
2. Read out the file with 272 initial labelled data points
3. Divided the data set into labeled (272) and unlabeled instances (7405) => ~0.035% labeled instances
4. Data cleaning including mapping to shorter abbreviations of intents and a replacing NAN with UNK_UNK labels etc.
5. A function to split the labeled and unlabeled dataset into four files as follows:

- labeled.tsv: Contains all the labeled data points
- unlabeled.tsv: Contains all the unlabeled data points
- test.tsv: Contains the test utterances
- test_OOS.tsv: Contains the test out-of-space utterances

For each intent, there are 210 training utterances, and 62 testing utterances.

6. Installing Dependencies:

1. https://github.com/guillaumegenthial/tf_metrics.git
2. tensorflow-gpu version - 1.14.0
3. gast version - 0.2.2 (Generative adversarial solver trainer)

7. Training:

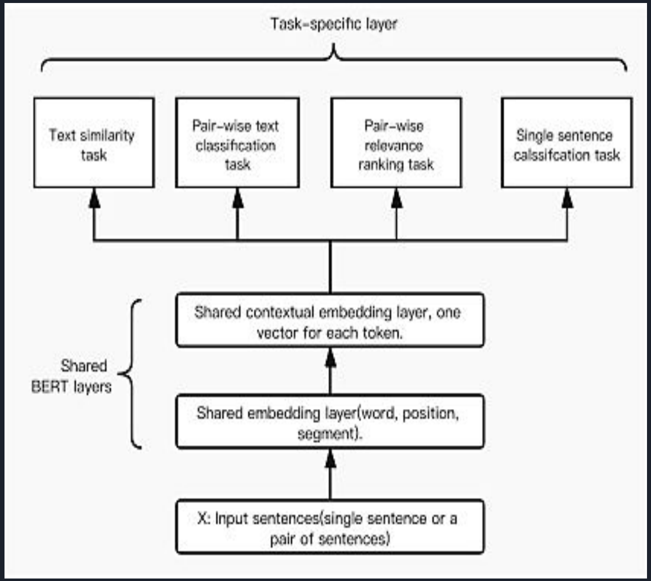
For simplicity, run a script file with the following parameters configured.

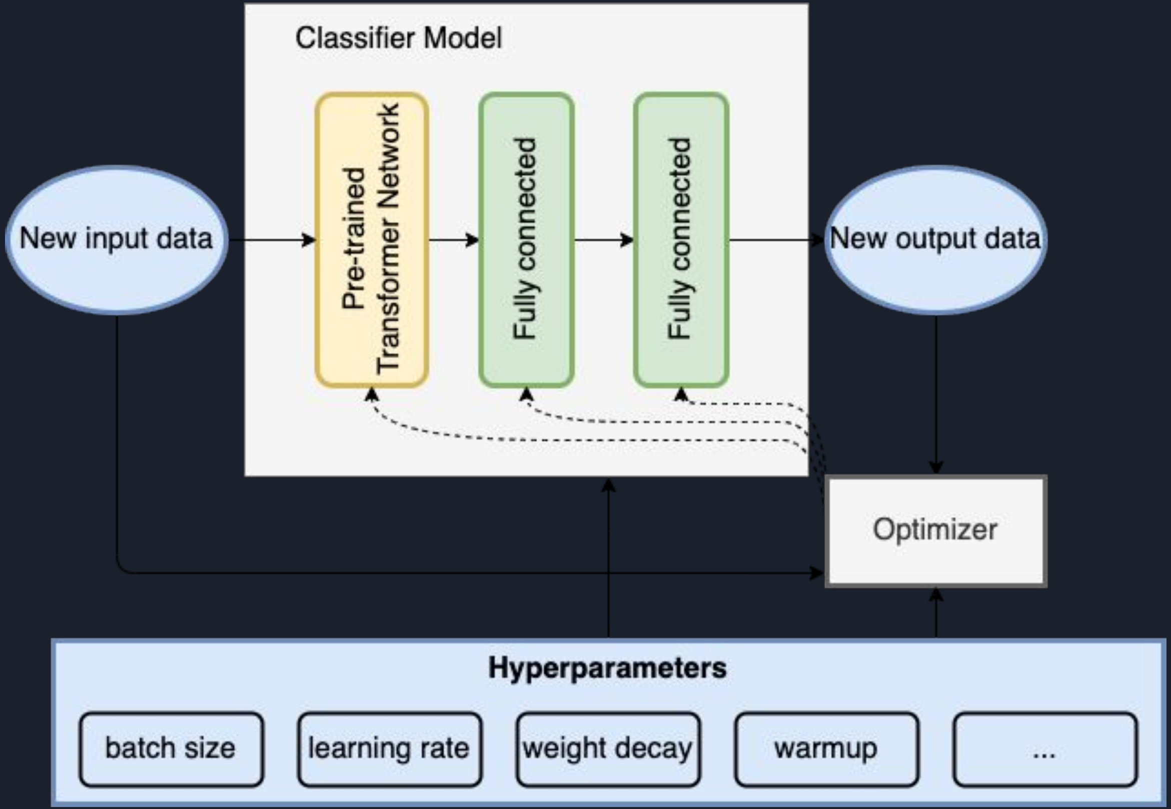
Final Hyperparameters used: (do_train=true, do_eval=true,do_predict=false, and pred_OOS=false)

1. learning rate = 2e-5
2. label_rate = 0.02
3. max_seq_length = 64
4. train_batch_size = 64
5. warmup_proportion = 0.1
6. epochs = 10

Configure data and output model directories path.

Model Architecture


Task-specific layer
Text similarity task
Pair-wise text classification task
Pair-wise relevance ranking task
Single sentence calssifcation task
Shared BERT layers
Shared contextual embedding layer, one vector for each token.
Shared embedding layer(word, position, segment).
X: Input sentences(single sentence or a pair of sentences)



What goes into it?

1. Let us assume we are facing a sentence classification task over k categories. Given an input sentence s = (t1, ..., tn) BERT produces in output n + 2 vector representations in Rd , i.e., (hCLS, ht1 , ..., htn , hSEP)

2. We add on top of BERT the SS-GAN architecture by introducing
   i) a discriminator D for classifying examples, and ii) a generator G acting adversarially

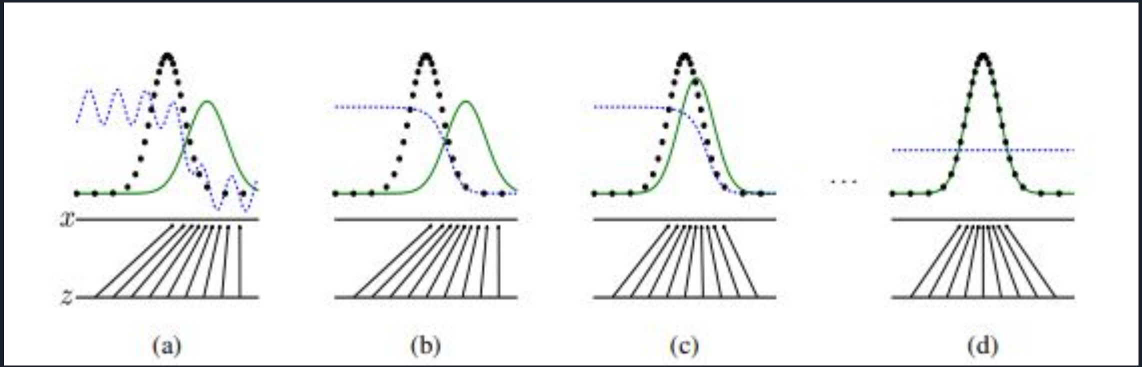
- G is a Multi Layer Perceptron (MLP) that takes in input a 100-dimensional noise vector drawn from N(μ, σ2) (standard normal distribution) and produces in output a vector hfake ∈ Rd .
- The discriminator is another MLP that receives in input a vector h∗ ∈ Rd. h∗ can be either hfake produced by the generator or hCLS for unlabeled or labeled examples from the real distribution. The last layer of D is a softmax-activated layer, whose output is a k + 1 vector of logits.

How it works?

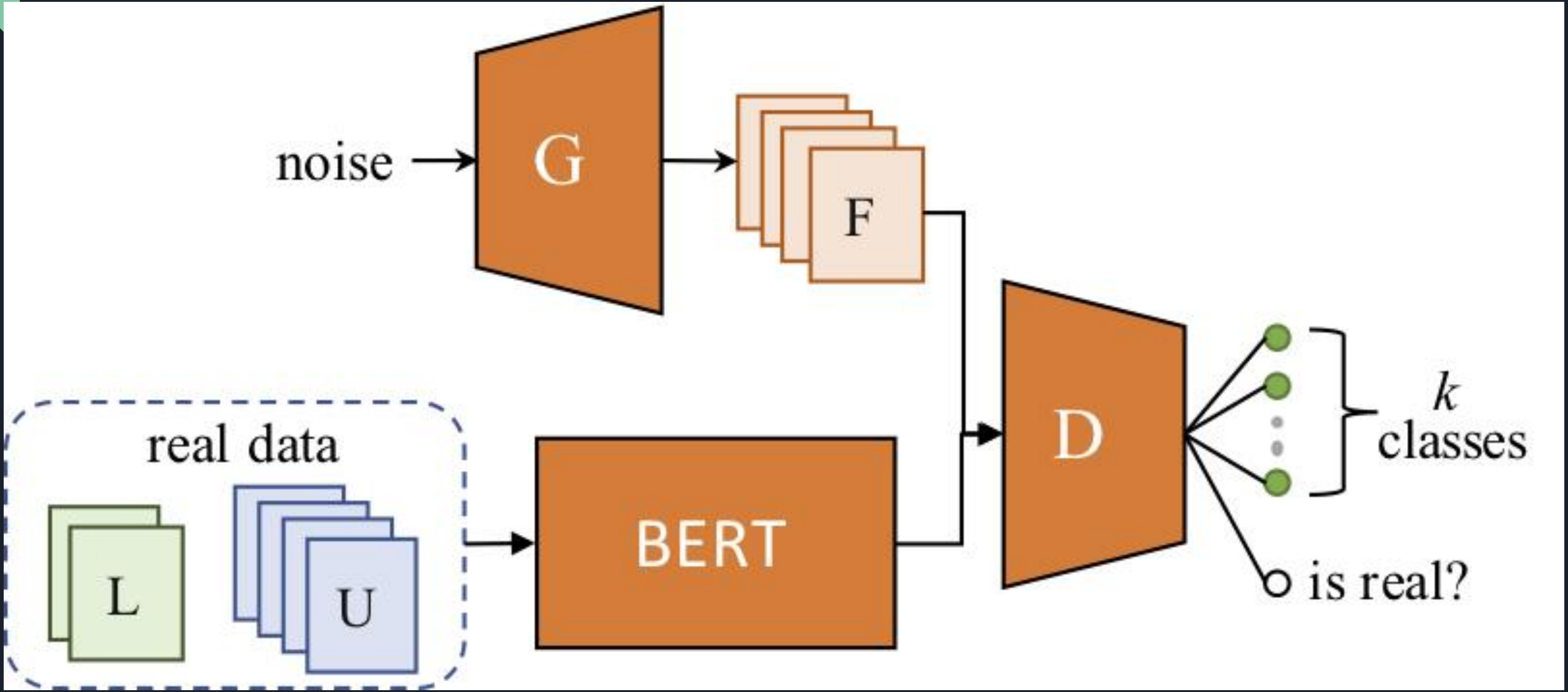
- During the forward step, when real instances are sampled (i.e., h∗ = hCLS), D should classify them in one of the k categories; but when h∗ = hfake, it should classify each example in the k + 1 category.

- The training process tries to optimise between the generator loss and the discriminator loss i.e. We train D to maximize the probability of assigning the correct label to both training examples and samples from G. We simultaneously train G to minimize log(1 − D(G(z)))

where, G and D area differentiable functions represented by a multilayer perceptron
pz(z) is a prior on input noise variable,



Model Architecture



Final Loss and Output

LD = LDsup. + LDunsup.

LDsup. measures the error in assigning the wrong class to a real example among the original k categories.

LDunsup. measures the error in incorrectly recognizing a real (unlabeled) example as fake and not recognizing a fake example

where: $LDsup.=-(Ex,y\sim pd \log[pm(\hat{y} = y|x, y \in (1, ..., k))])$
 $LDunsup.=-Ex\sim pd \log[1- pm (\hat{y} = y|x, y = k+ 1)] - Ex\sim G \log [pm(\hat{y} = y|x, y = k + 1)]$

Given $pm(\hat{y} = y|x, y = k + 1)$ the probability provided by the model m that a generic example x is associated with the fake class and $pm(\hat{y} = y|x, y \in (1, ..., k))$ that x is real; thus belonging to one of the labeled classes

Final Loss and Output

LG = Feature matching. + LGunsup.

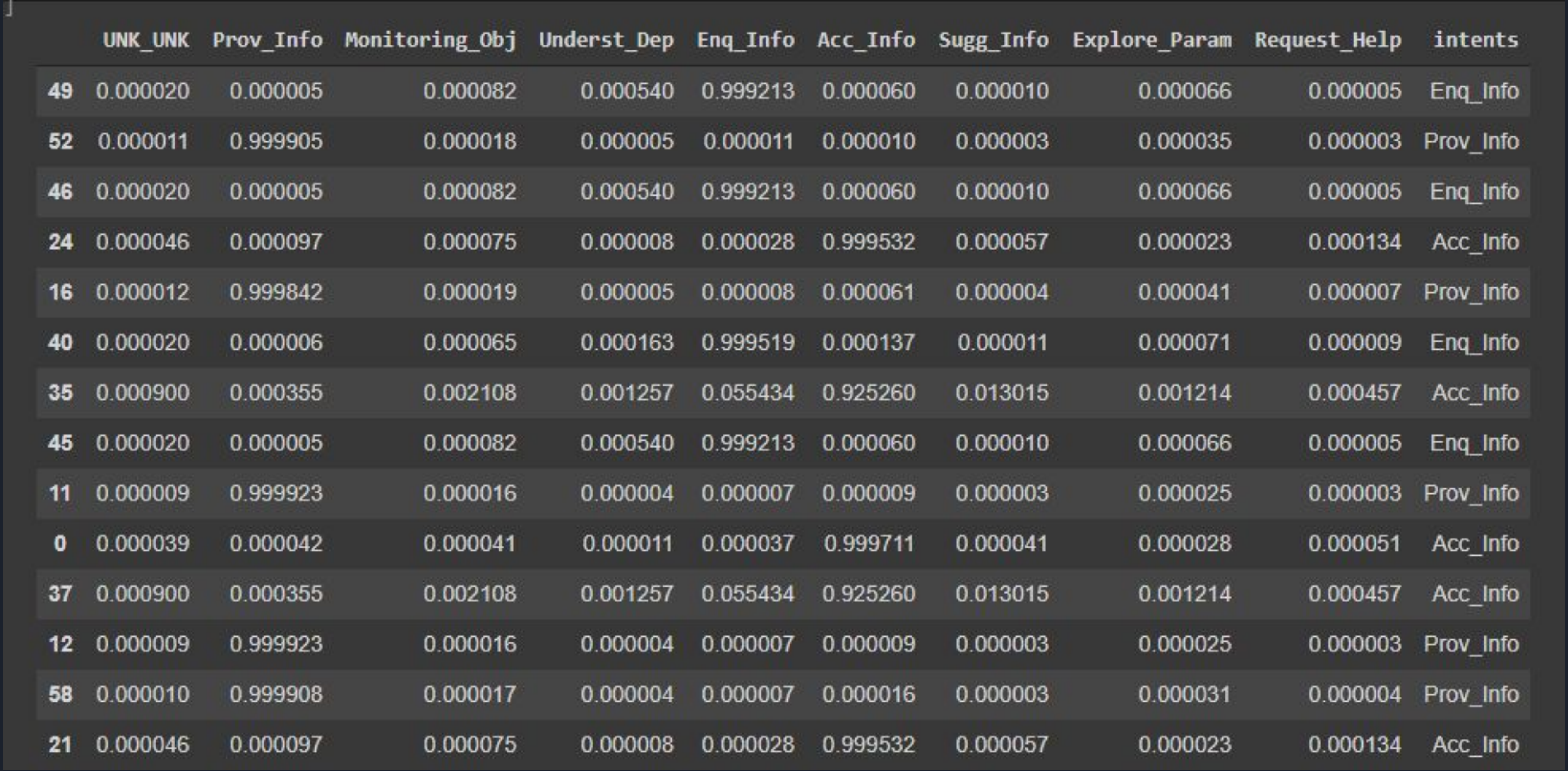
G is expected to generate examples that are similar to the ones sampled from the real distribution pd

The feature matching loss of G is then defined as:

LGfeature matching= ||Ex ~ pd f(x) − Ex ~ Gf(x)||^2

(f(x) denote the activation on an intermediate layer of D.

The G loss also considers the error induced by fake examples correctly identified by D, i.e.,
LGunsup.=−Ex~G log[1− pm(ˆy = y|x,y = k+ 1)]



	UNK_UNK	Prov_Info	Monitoring_Obj	Underst_Dep	Enq_Info	Acc_Info	Sugg_Info	Explore_Param	Request_Help	intents
49	0.000020	0.000005	0.000082	0.000540	0.999213	0.000060	0.000010	0.000066	0.000005	Enq_Info
52	0.000011	0.999905	0.000018	0.000005	0.000011	0.000010	0.000003	0.000035	0.000003	Prov_Info
46	0.000020	0.000005	0.000082	0.000540	0.999213	0.000060	0.000010	0.000066	0.000005	Enq_Info
24	0.000046	0.000097	0.000075	0.000008	0.000028	0.999532	0.000057	0.000023	0.000134	Acc_Info
16	0.000012	0.999842	0.000019	0.000005	0.000008	0.000061	0.000004	0.000041	0.000007	Prov_Info
40	0.000020	0.000006	0.000065	0.000163	0.999519	0.000137	0.000011	0.000071	0.000009	Enq_Info
35	0.000900	0.000355	0.002108	0.001257	0.055434	0.925260	0.013015	0.001214	0.000457	Acc_Info
45	0.000020	0.000005	0.000082	0.000540	0.999213	0.000060	0.000010	0.000066	0.000005	Enq_Info
11	0.000009	0.999923	0.000016	0.000004	0.000007	0.000009	0.000003	0.000025	0.000003	Prov_Info
0	0.000039	0.000042	0.000041	0.000011	0.000037	0.999711	0.000041	0.000028	0.000051	Acc_Info
37	0.000900	0.000355	0.002108	0.001257	0.055434	0.925260	0.013015	0.001214	0.000457	Acc_Info
12	0.000009	0.999923	0.000016	0.000004	0.000007	0.000009	0.000003	0.000025	0.000003	Prov_Info
58	0.000010	0.999908	0.000017	0.000004	0.000007	0.000016	0.000003	0.000031	0.000004	Prov_Info
21	0.000046	0.000097	0.000075	0.000008	0.000028	0.999532	0.000057	0.000023	0.000134	Acc_Info

Results on test data:

For 61 instances in the test set, we got the following results:

- Accuracy: 0.23 (weighted average for all the labels)
- Precision: 0.19
- Recall: 0.23
- F1-score: 0.21

F1 Score might be a better measure to use if we need to seek a balance between Precision and Recall especially when we have an uneven class distribution.

Final Results on unseen data:

The script will first download the BERT-based model, and then it will run the experiments. After some time, there will be a file in the output: *qc-fine_statistics_GANBERT0.02.txt*.

This contains the performance measures of GANBERT, after training GAN-BERT on only 211 labeled examples in a classification task involving 8 classes, the following results are obtained:

Model converged at d_loss = 1.82883 and g_loss = 0.31207836 at 10 epochs

**** Final Performance Parameters: ****

1. eval_accuracy = 0.704918
2. eval_f1_macro = 0.27156514
3. eval_f1_micro = 0.704918
4. eval_loss = 2.140911 (g_loss + d_loss)
5. eval_precision = 0.704918
6. eval_recall = 0.704918

Scope of the Project/Research/Review Paper:

1. From McGrath theory of group behavior, the focus on model of operation would be:

- Inception
- Problem-solving
- Conflict resolution
- Execution; based on the conversation context and intent

Or by classifying teamwork dimensions

- Coordination
- Mutual Performance Monitoring
- Team Decision Making
- Constructive Conflict
- Team emotional support
- Team Commitment

The model can make the teams aware of the deficiencies during different phases of design task.

Method 2 - Under Semi-Supervised Intent Classification:

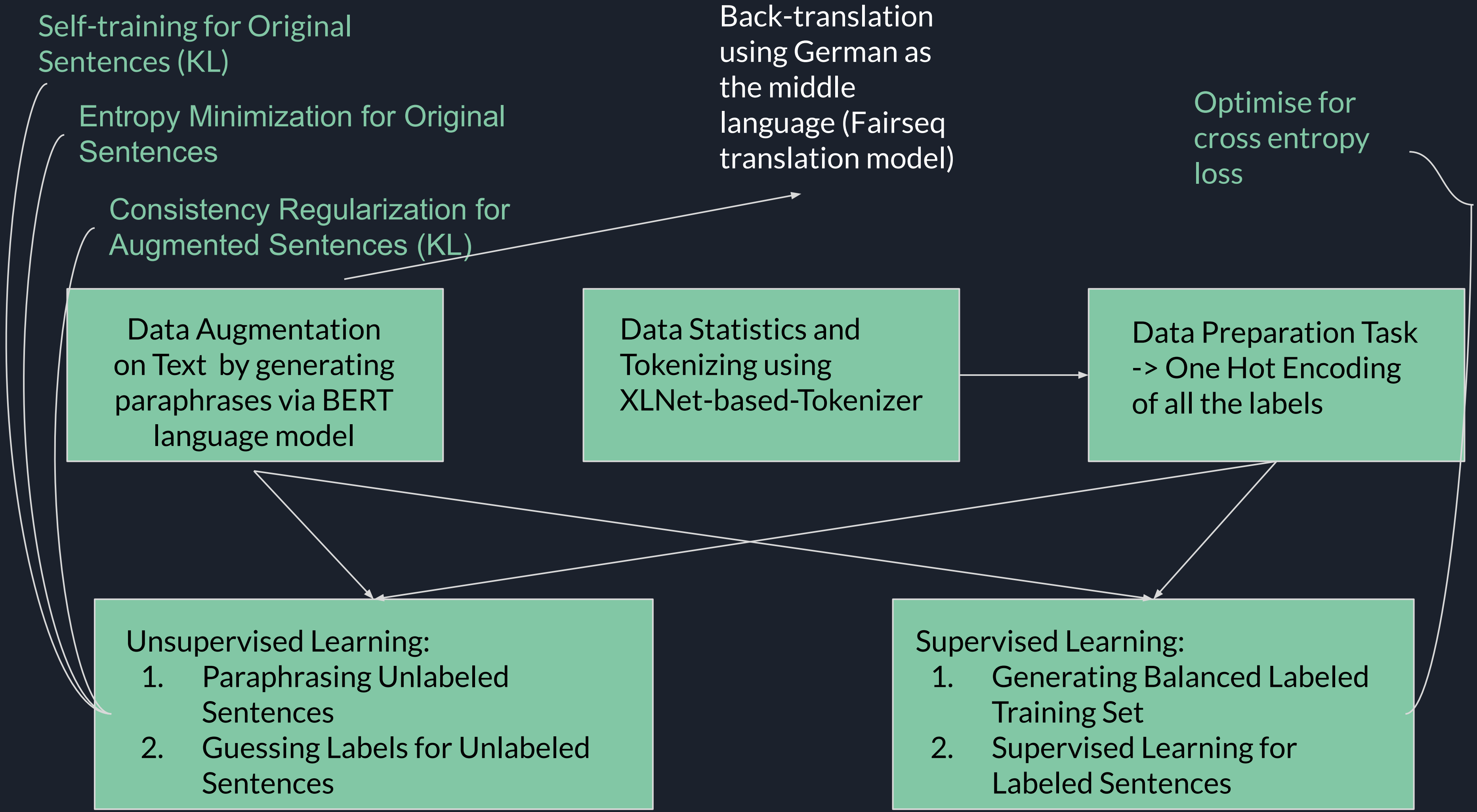
-> Self Learning with Data Augmentation Technique

SMDA utilizes recent transformer-based models to encode each sentence and employs back translation techniques to paraphrase given sentences as augmented data.

-> For labeled sentences, we will perform data augmentations to uniform the label distributions and computed supervised loss during training process.

-> For unlabeled sentences, we will explore self-training by regarding low-entropy predictions over unlabeled sentences as pseudo labels, assuming high-confidence predictions as labeled data for training. (using model.predict_proba)

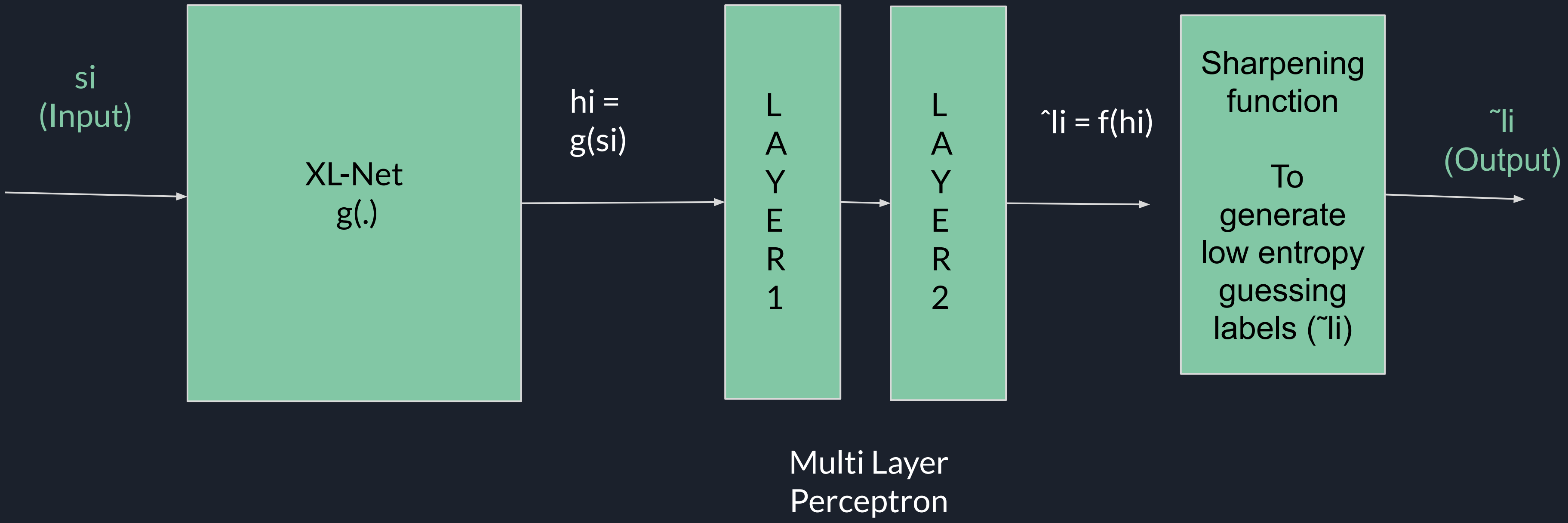
-> Final metric to consider would be F1-score


Self-training for Original Sentences (KL)
Entropy Minimization for Original Sentences
Consistency Regularization for Augmented Sentences (KL)
Back-translation using German as the middle language (Fairseq translation model)
Optimise for cross entropy loss
Data Augmentation on Text by generating paraphrases via BERT language model
Data Statistics and Tokenizing using XLNet-based-Tokenizer
Data Preparation Task -> One Hot Encoding of all the labels
Unsupervised Learning:
1. Paraphrasing Unlabeled Sentences
2. Guessing Labels for Unlabeled Sentences
Supervised Learning:
1. Generating Balanced Labeled Training Set
2. Supervised Learning for Labeled Sentences

Modeling:

For each input labeled sentence si ,

1. We will use XLNet -> g(.) to encode it into hidden representation hi = g(si), then
2. Will pass them through a 2-layer MLP to predict the class distribution ˆli = f(hi)



Semi-Supervised Objective Function:

We will combine the supervised and unsupervised learning described above to form our overall semi-supervised objective function:

L = Supervised Loss + γ (Unsupervised Loss)

where γ is the balanced weight between supervised and unsupervised loss term. (0-1)

Note:
Supervised Loss = LS(si , li) = − Summation(li log f(g(si)))
Unsupervised Loss =

NLP Conferences:

1. ACL: Association for Computational Linguistics
2. EMNLP: Empirical Methods in Natural Language Processing
3. NAACL: North American Chapter of the Association for Computational Linguistics
4. EACL: European Chapter of the Association for Computational Linguistics
5. COLING: International Conference on Computational Linguistics
6. CoNLL: Conference on Natural Language Learning

Other Conferences:

1. ACM CHI Conference: Human Factors in Computing Systems
2. International Conference on Computer-Aided Systems

Thank you!