A Project Report
on

Politeness Transfer: A Tag and Generate Approach

submitted in partial fulfillment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

17WH1A0561	Ms. P. KEERTHI
17WH1A0574	Ms. M. SATHWIKA
17WH1A05B5	Ms. A. SHIVANI

under the esteemed guidance of

Mr. Bapiraju Mudunuri

Assistant Professor

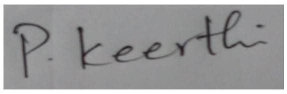
VISHNU
UNIVERSAL LEARNING

Department of Computer Science and Engineering
BVRIT HYDERABAD
College of Engineering for Women
(NBA Accredited – EEE, ECE, CSE and IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

June, 2020

DECLARATION

We hereby declare that the work presented in this project entitled **"Politeness Transfer: A Tag and Generate Approach"** submitted towards completion of Project Work in IV year of B.Tech, CSE at 'BVRIT HYDERABAD College of Engineering For Women', Hyderabad is an authentic record of our original work carried out under the guidance of Mr. Bapiraju Mudunuri, Assistant Professor, Department of CSE.



Sign. with date:

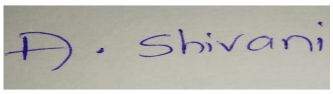
Ms. P. KEERTHI

(17WH1A0561)

Sign. with date:

Ms. M. SATHWIKA

(17WH1A0574)



Sign. with date:

Ms. A. SHIVANI

(17WH1A05B5)

BVRIT HYDERABAD
College of Engineering for Women
(NBA Accredited – EEE, ECE, CSE and IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

Department of Computer Science and Engineering

Certificate

This is to certify that the Project Work report on "**Politeness Transfer: A Tag and Generate Approach**" is a bonafide work carried out by Ms. P. KEERTHI (17WH1A0561); Ms. M. SATHWIKA (17WH1A0574); Ms. A. SHIVANI (17WH1A05B5) in the partial fulfillment for the award of B.Tech. degree in **Computer Science and Engineering, BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Head of the Department	**Guide**
Dr. K. Srinivasa Reddy	**Mr. Bapiraju Mudunuri**
Professor, CSE	**Asst. Professor, CSE**

External Examiner

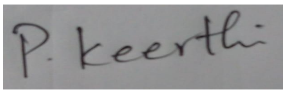
Acknowledgements

We would like to express our sincere thanks to **Dr. K V N Sunitha, Principal**, **BVRIT HYDERABAD College of Engineering for Women**, for providing the working facilities in the college.

Our sincere thanks and gratitude to our **Dr. K. Srinivasa Reddy, Professor**, Department of CSE, **BVRIT HYDERABAD College of Engineering for Women** for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Mr. Bapiraju Mudunuri, Assistant Professor**, Department of CSE**, BVRIT HYDERABAD College of Engineering for Women** for his constant guidance, encouragement and moral support throughout the project.

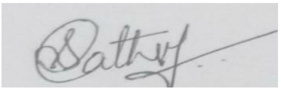
Finally, we would also like to thank our Project Coordinator, all the faculty and staff of **CSE** Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.



Sign. with date:

Ms. P. KEERTHI

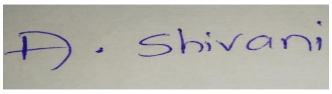
(17WH1A0561)



Sign. with date:

Ms. M. SATHWIKA

(17WH1A0574)



Sign. with date:

Ms. A. SHIVANI

(17WH1A05B5)

Contents

ABSTRACT

It is a task of politeness transfer which involves converting non-polite sentences to polite sentences while preserving the meaning. We also provide a dataset of more than 1.39 million instances automatically labeled for politeness to encourage benchmark evaluations on this new task. We design a tag and generate a pipeline that identifies stylistic attributes and subsequently generates a sentence in the target style while preserving most of the source content. For politeness as well as five other transfer tasks, our model outperforms the state-of-the-art methods on automatic metrics for content preservation, with a comparable or better performance on style transfer accuracy. Additionally, our model surpasses existing methods on human evaluations for grammaticality, meaning preservation and transfer accuracy across all the six style transfer tasks.

LIST OF FIGURES

1. INTRODUCTION

Politeness plays a crucial role in social interaction, and is closely tied with power dynamics, social distance between the participants of a conversation, and gender. It is also imperative to use the appropriate level of politeness for smooth communication in conversations, organizational settings like emails, memos, official documents, and many other settings. Notably, politeness has also been identified as an interpersonal style which can be decoupled from content. Motivated by its central importance, in this paper we study the task of converting non-polite sentences to polite sentences while preserving the meaning.

1.1 Objectives

In this Project, we convert non-polite sentences to polite sentences while preserving the meaning. We focus on the task of transferring the non-polite sentences to polite sentences, where we simply define non-politeness to be the absence of both politeness and impoliteness.

1.2 Methodology

We are given non-parallel samples of sentences X1 = {x (1) 1 . . . x (1) n } and X2 = {x (2) 1 . . . x (2) m } from styles S1 and S2 respectively. The objective of the task is to efficiently generate samples Xˆ 1 = {ˆx (2) 1 . . . ˆx (2) n } in the target style S2, conditioned on samples in X1. For a style Sv where v ∈ {1, 2}, we begin by learning a set of phrases (Γv) which characterize the style Sv. The presence of phrases from Γv in a sentence xi would associate the sentence with the style Sv. For example, phrases like “pretty good” and “worth every penny” are characteristic of the “positive” style in the case of sentiment transfer tasks.

We propose a two-stage approach where we first infer a sentence z(xi) from x (1) i using a model, the tagger. The goal of the tagger is to ensure that the sentence z(xi) is agnostic to the original style (S1) of the input sentence. Conditioned on z(xi), we then generate the transferred sentence ˆx (2) i in the target style S2 using another model, the generator.

1.3 Dataset

The dataset we are using is Enron corpus. It consists of a large set of email conversations exchanged by the employees of the Enron corporation. Emails serve as a medium for exchange of requests, serving as an ideal application for politeness transfer. We begin by pre-processing the raw Enron corpus. The first set of pre-processing steps and de-duplication yielded a corpus of roughly 2.5 million sentences. Further pruning led to a cleaned corpus of over 1.39 million sentences. Finally, we use a politeness classifier to assign politeness scores to these sentences and filter them into ten buckets based on the score. All the buckets are further divided into train, test, and dev splits.

Creation

- The original dataset is located at: https://www.cs.cmu.edu/~./enron/
- As discussed in the paper, the following steps were followed to create the dataset:
  - Pre-processing: Tokenization (done using spacy) and conversion to lowercase.
  - We further prune the corpus by removing the sentences that: were less than 3 words long, had more than 80% numeri-cal tokens, contained email addresses, or had repeated occurrences of spurious characters.

1	comment	final_agreed_rating	target	stanford_tool_score
2	I removed webdav plugin and tried to do sync up on the repo , I got "Error: No repository found in path, or configured plugin not installed. Use 'syinit' to create one." Maybe it's better to show which plugin is missing and provide user with a command to install it (i.e. sy plugin install webdav -- snapshot)	polite	1	0.561259115
3	Could not load an SSH config at work today. Something is wrong with	neutral	0	0.393322534
4	oh, forgot to clarify: i'm only talking about this part: ((resizedImage->imageData + resizedImage->widthStep*x)[y]); so, if i'm using a cv::Mat here instead, i'd write: resizedImage.at<uchar>(x,y); and at least get the same resulting landmarks	neutral	0	0.534475941
5	This was a sync with master that disabled kafka due to build breakage	neutral	0	0.406828814
6	See this solution. It works without hacking the control. #1513"	neutral	0	0.439007419
7	@brandonarbiter I don't believe it's a requirement for pilot because	polite	1	0.387320355

Fig 1.3.1: Dataset

2. THEORETICAL ANALYSIS OF THE PROPOSED PROJECT

2.1 Requirements Gathering

2.1.1 Software Requirements

Programming Language	: Python 3.6
Dataset	: Dataset
Packages	:Tensorflow, Numpy, Pandas, Sklearn
Tool	: Spyder

2.1.2 Hardware Requirements

Operating System	: Windows 10
Processor	: Intel Core i5
Memory	: 8 GB (RAM)

2.2 Technologies Description

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Enron corpus Dataset

It consists of a large set of email conversations exchanged by the employees of the Enron corporation. Emails serve as a medium for exchange of requests, serving as an ideal application for politeness transfer.

Tensorflow

Transform makes extensive use of TensorFlow Transform for performing feature engineering on your dataset. TensorFlow Transform is a great tool for transforming feature data before it goes to your model and as a part of the training process. Common feature transformations include:

- **Embedding**: converting sparse features (like the integer IDs produced by a vocabulary) into dense features by finding a meaningful mapping from high-dimensional space to low dimensional space. See the Embeddings unit in the Machine-learning Crash Course for an introduction to embeddings.
- **Vocabulary generation**: converting strings or other non-numeric features into integers by creating a vocabulary that maps each unique value to an ID number.
- **Normalizing values**: transforming numeric features so that they all fall within a similar range.
- **Bucketization**: converting continuous-valued features into categorical features by assigning values to discrete buckets.
- **Enriching text features**: producing features from raw data like tokens, n-grams, entities, sentiment, etc., to enrich the feature set.

TensorFlow Transform provides support for these and many other kinds of transformations:

- Automatically generate a vocabulary from your latest data.
- Perform arbitrary transformations on your data before sending it to your model. TensorFlow Transform builds transformations into the TensorFlow graph for your model so the same transformations are performed at training and inference time. You can define transformations that refer to global properties of the data, like the max value of a feature across all training instances.

Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

3. DESIGN

3.1 Introduction

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Once system requirements have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed, reviewed and documented. System design can be viewed from either a technical or project management perspective. From the technical point of view, design consists of four activities – architectural design, data structure design, interface design and procedural design.

3.1.1 Proposed approach:

fig 3.1 Proposed approach

3.2 Architecture Diagram

Web applications are by nature distributed applications, meaning that they are programs that run on more than one computer and communicate through network or server. Specifically, web applications are accessed with a web browser and are popular because of the ease of using the browser as a user client. For the enterprise, software on potentially thousands of client computers is a key reason for their popularity. Web applications are used for web mail, online retail sales, discussion boards, weblogs, online banking, and more. One web application can be accessed and used by millions of people.

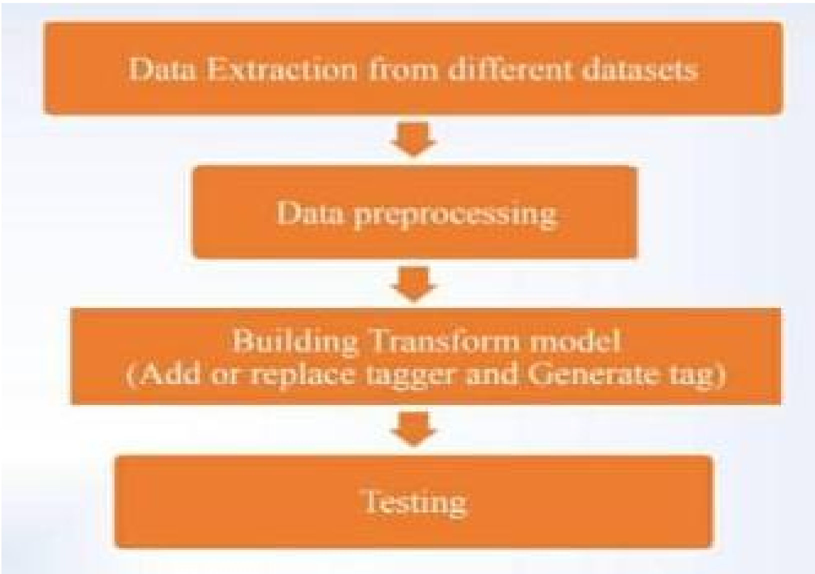


Fig 3.2: Architecture Diagram

3.3 UML Diagrams

3.3.1 Use Case Diagram

To model a system, the most important aspect is to capture the dynamic behavior. Dynamic behavior means the behavior of the system when it is running/operating.

Only static behavior is not sufficient to model a system; rather dynamic behavior is more important than static behavior. In UML, there are five diagrams available to model the dynamic nature and use case diagrams are one of them. Now as we have to discuss that the use case diagram is dynamic in nature, there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. Use case diagrams consist of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

Hence to model the entire system, a number of use case diagrams are used.

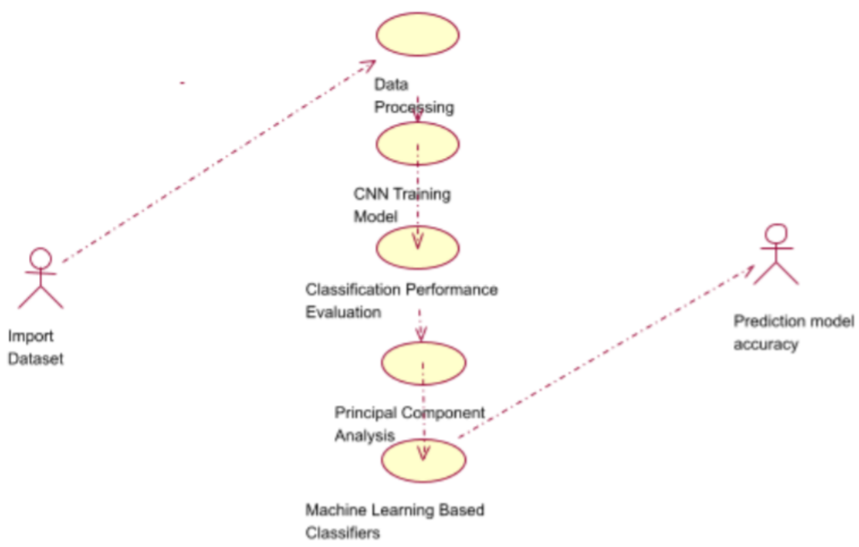


Fig 3.3.1: Use Case Diagram

3.3.2 Sequence Diagram

Sequence Diagrams Represent the objects participating in the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.

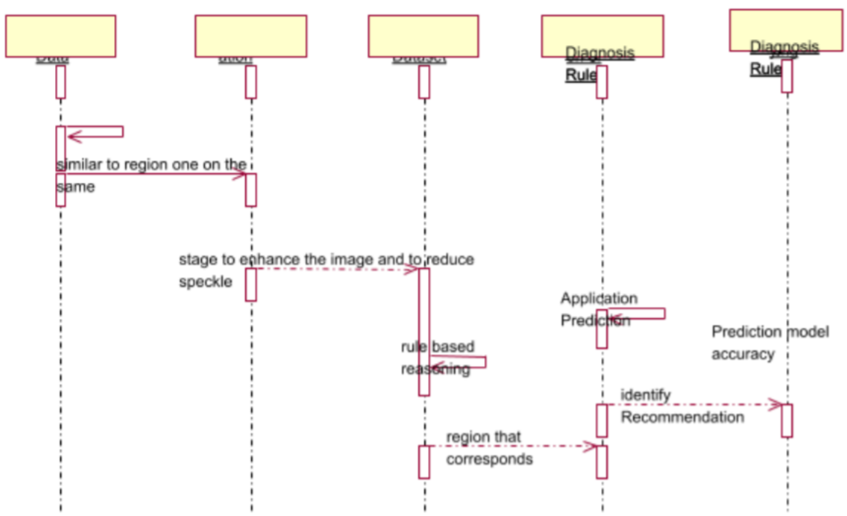


Fig 3.3.2: Sequence Diagram

4. IMPLEMENTATION

4.1 Coding

lstm.py

```
import tensorflow as tf
if tf.test.gpu_device_name():
    print('Default GPU Device: {}'.format(tf.test.gpu_device_name()))
else:
    print("Please install GPU version of TF")
import pandas as pd
data = pd.read_excel('politeness_2k_data.xlsx')
data = data.fillna('_NA_')

label_names = ["target"]
y_train = data[label_names].values

import numpy as np
data['doc_len'] = data['comment'].apply(lambda words: len(words.split(" ")))
max_seq_len = np.round(data['doc_len'].mean() + data['doc_len'].std()).astype(int)

from tqdm import tqdm
import nltk
from nltk.tokenize import word_tokenize



from nltk.corpus import stopwords
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
stop_words.update(['.', ',', '"', "'", ':', ';', '(', ')', '[', ']', '{', '}'])



## preprocessing starting
raw_docs_train = data['comment'].tolist()
```

```
num_classes = len(label_names)

print("pre-processing train data...")

processed_docs_train = []
for doc in tqdm(raw_docs_train):
    tokens = word_tokenize(doc)
    filtered = [word for word in tokens if word not in stop_words]
    processed_docs_train.append(" ".join(filtered))

from tensorflow import keras
from tensorflow.keras.preprocessing import sequence

MAX_NB_WORDS = 10000
from tensorflow.keras.preprocessing.text import Tokenizer



print("tokenizing input data...")
tokenizer = Tokenizer(num_words=MAX_NB_WORDS, lower=True, char_level=False)
tokenizer.fit_on_texts(processed_docs_train )  #leaky
word_seq_train = tokenizer.texts_to_sequences(processed_docs_train)
word_index = tokenizer.word_index
print("dictionary size: ", len(word_index))

#pad sequences
word_seq_train = sequence.pad_sequences(word_seq_train, maxlen=max_seq_len)

print('loading word embeddings...')
import os, re, csv, math, codecs

embeddings_index = {}
f = codecs.open('Embedding/crawl-300d-2M.vec', encoding='utf-8')
```

```
for line in tqdm(f):
    values = line.rstrip().rsplit(' ')
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()

print('found %s word vectors' % len(embeddings_index))

#embedding matrix

print('preparing embedding matrix...')
embed_dim = 300
words_not_found = []
nb_words = min(MAX_NB_WORDS, len(word_index)+1)
embedding_matrix = np.zeros((nb_words, embed_dim))

for word, i in word_index.items():
    if i >= nb_words:
        continue
    embedding_vector = embeddings_index.get(word)
    if (embedding_vector is not None) and len(embedding_vector) > 0:
        # words not found in embedding index will be all-zeros.
        embedding_matrix[i] = embedding_vector
    else:
        words_not_found.append(word)
print('number of null word embeddings: %d' % np.sum(np.sum(embedding_matrix, 
axis=1) == 0))

import tensorflow as tf
from sklearn.model_selection import KFold
kf = KFold(n_splits=10, shuffle=True, random_state=125)
from tensorflow.keras.callbacks import EarlyStopping
#from matplotlib import pyplot
```

```
from sklearn.model_selection import train_test_split
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import EarlyStopping

from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import  precision_score
from sklearn.metrics import  f1_score

#only LSTM
from tensorflow.keras import regularizers

from tensorflow.keras.layers import BatchNormalization
import tensorflow as tf



#max_features =22248
#nb_words=22248
embedding_dim =300
sequence_length = 100

def LSTM_model():
   model =  keras.Sequential()
          #model.add(tf.keras.layers.Embedding(max_features +1, embedding_dim,
input_length=sequence_length,\
                          #embeddings_regularizer = regularizers.l2(0.005)))
                                                                    model.add(
keras.layers.Embedding(nb_words,embed_dim,input_length=max_seq_len,weights=[
embedding_matrix],trainable=False))
   model.add( keras.layers.Dropout(0.4))

                          model.add(        keras.layers.LSTM(embedding_dim,dropout=0.2,
recurrent_dropout=0.2,return_sequences=True,\
```

```
                                          kernel_regularizer=regularizers.l2(0.005),\
                                          bias_regularizer=regularizers.l2(0.005)))

   model.add( keras.layers.Flatten())

   model.add( keras.layers.Dense(512, activation='relu',\
                        kernel_regularizer=regularizers.l2(0.001),\
                        bias_regularizer=regularizers.l2(0.001),))
   model.add( keras.layers.Dropout(0.4))

   model.add( keras.layers.Dense(8, activation='relu',\
                        kernel_regularizer=regularizers.l2(0.001),\
                        bias_regularizer=regularizers.l2(0.001),))
   model.add( keras.layers.Dropout(0.4))


   model.add( keras.layers.Dense(1,activation='sigmoid'))


model.compile(loss=tf.keras.losses.BinaryCrossentropy(),optimizer=tf.keras.optimize
rs.Adam(1e-3),metrics=['acc'])

   model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
   return model

LSTM_model().summary()

from tensorflow.keras.utils import plot_model

#plot_model(LSTM_model(), to_file='LSTMmodel.png', show_shapes=True,
show_layer_names=True)

es_callback = EarlyStopping(monitor='val_loss', patience=3)
```

```
lstm_run_precision = []
lstm_run_recall = []
lstm_run_f1score = []
lstm_run_accuracy = []

count = 1
num_epochs = 40

for train_index, test_index in kf.split(word_seq_train):
    x_trn, x_tst = word_seq_train[train_index], word_seq_train[test_index]
    y_trn, y_tst = y_train[train_index], y_train[test_index]

        x_new_train, x_val, y_new_train, y_val= train_test_split(x_trn, y_trn,
test_size=0.11115, random_state=125)

    print("\nFold ", count)
    lstm_model=LSTM_model()



    es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)

    history =lstm_model.fit( x_new_train, y_new_train, batch_size=256,
          epochs=num_epochs, validation_data=(x_val, y_val), callbacks=[es_callback],
shuffle=False)

    _, train_acc =lstm_model.evaluate(x_new_train,  y_new_train, verbose=0)
    _, val_acc =  lstm_model.evaluate(x_val, y_val, verbose=0)
    print('Train: %.3f, Test: %.3f' % (train_acc, val_acc))



    #plt.savefig('LSTM with fasttext SE data accuracy graph.png')
    #plt.show()
```

```
    y_pred = lstm_model.predict(x_tst)
    y_pred = (y_pred >= 0.5)


    from sklearn import metrics
    print(metrics.classification_report(y_tst, y_pred))

    lstm_precision = precision_score(y_tst, y_pred, pos_label=1)
    lstm_recall = recall_score(y_tst, y_pred, pos_label=1)
    lstm_f1score = f1_score(y_tst, y_pred, pos_label=1)
    lstm_accuracy = accuracy_score(y_tst, y_pred)

    lstm_run_accuracy.append(lstm_accuracy)
    lstm_run_f1score.append(lstm_f1score)
    lstm_run_precision.append(lstm_precision)
    lstm_run_recall.append(lstm_recall)

    count = count+1

import tensorflow
from tensorflow.keras.models import Model

from tensorflow.keras.layers import Dense, Input, LSTM,GlobalMaxPool1D
maxlen=max_seq_len
embed_size=300
max_features=nb_words
def Bi_LSTM_base():
    inp = keras.layers.Input(shape=(maxlen,))
            x = tensorflow.keras.layers.Embedding(max_features, embed_size,
weights=[embedding_matrix])(inp)
        x = tensorflow.keras.layers.Bidirectional(tensorflow.keras.layers.LSTM(50,
return_sequences=True, dropout=0.1, recurrent_dropout=0.1))(x)
```

```
    x = tensorflow.keras.layers.GlobalMaxPool1D()(x)
    x = tensorflow.keras.layers.Dense(50, activation="relu")(x)
    x = tensorflow.keras.layers.Dropout(0.1)(x)
    x = tensorflow.keras.layers.Dense(1, activation="sigmoid")(x)
    model = Model(inputs=inp, outputs=x)
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
Bi_LSTM_base().summary()
blbase_run_precision = []
blbase_run_recall = []
blbase_run_f1score = []
blbase_run_accuracy = []

count = 1

for train_index, test_index in kf.split(word_seq_train):
    x_trn, x_tst = word_seq_train[train_index], word_seq_train[test_index]
    y_trn, y_tst = y_train[train_index], y_train[test_index]

    x_new_train, x_val, y_new_train, y_val= train_test_split(x_trn, y_trn,
test_size=0.11115, random_state=125)

    print("\nFold ", count)
    bilstmbase_model=Bi_LSTM_base()

    #model_lstm_fasttext=model_with_embedding()

    es = EarlyStopping(monitor='val_loss', mode='min', verbose=1)

    history =bilstmbase_model.fit( x_new_train, y_new_train, batch_size=32,
          epochs=num_epochs, validation_data=(x_val, y_val), callbacks=[es_callback],
shuffle=False)

    _, train_acc = bilstmbase_model.evaluate(x_new_train,  y_new_train, verbose=0)
```

```
    _, val_acc =  bilstmbase_model.evaluate(x_val, y_val, verbose=0)
    print('Train: %.3f, Test: %.3f' % (train_acc, val_acc))



    y_pred = bilstmbase_model.predict(x_tst)
    y_pred = (y_pred >= 0.5)



    from sklearn import metrics
    print(metrics.classification_report(y_tst, y_pred))

    blbase_precision = precision_score(y_tst, y_pred, pos_label=1)
    blbase_recall = recall_score(y_tst, y_pred, pos_label=1)
    blbase_f1score = f1_score(y_tst, y_pred, pos_label=1)
    blbase_accuracy = accuracy_score(y_tst, y_pred)

    blbase_run_accuracy.append(blbase_accuracy)
    blbase_run_f1score.append(blbase_f1score)
    blbase_run_precision.append(blbase_precision)
    blbase_run_recall.append(blbase_recall)

    count = count+1
```

estimation.py

```
import csv

labels = ['ID', 'Message', 'NS', 'NNS']
filenames = ["BinaryLabeling.csv", "StrongNeutralLabeling.csv",
             "WeakNeutralLabeling.csv", "IntermediateLabeling.csv",
            "PartitionsLabeling.csv"]
fileobjs = [open("LabeledData/" + i, "r") for i in filenames]
readers = [csv.reader(i) for i in fileobjs]
```

```
from nltk.tokenize import word_tokenize
from nltk import NaiveBayesClassifier
from nltk.classify import accuracy
from collections import Counter

# Create featureset from all individual words in training
next(readers[0], None)
num_train = 900 # Training comes from first 900 of 1000 samples
all_words = set()
for row in readers[0]:
    if num_train <= 0:
        break;
    line = word_tokenize(row[1])
    for word in line:
        all_words.add(word)
    num_train -= 1

# Using seek(0) resets reader
fileobjs[0].seek(0)

def bag_of_words(sentence):
    d = dict.fromkeys(all_words, 0)
    c = Counter(word_tokenize(sentence))
    for i in c:
        d[i] = c[i]
    return d

NB_classifiers_NS = []
NB_classifiers_NNS = []
NB_tests_NS = []
NB_tests_NNS = []
for i in readers:
    next(i, None)
```

```
    all_data = list(i)
    train_NS = [(bag_of_words(row[1]), row[2]) for row in all_data[:850]]
    train_NNS = [(bag_of_words(row[1]), row[3]) for row in all_data[:850]]
    NB_tests_NS.append([(bag_of_words(row[1]), row[2]) for row in all_data[850:]])
             NB_tests_NNS.append([(bag_of_words(row[1]),    row[3])    for    row    in
all_data[850:]])

    NB_classifiers_NS.append(NaiveBayesClassifier.train(train_NS))
    NB_classifiers_NNS.append(NaiveBayesClassifier.train(train_NNS))

for i in range(len(filenames)):
    print(filenames[i])
    print("native speaker:")
    print(accuracy(NB_classifiers_NS[i], NB_tests_NS[i]))
    print("non-native speaker:")
    print(accuracy(NB_classifiers_NNS[i], NB_tests_NNS[i]))

import requests
import re
import textstat
import json
import time

# Variables for perspective API call
# headers and parameters for perspective api call
api_key = 'AIzaSyBaMPpybrBfyWF54hvkFK1QuEBPPKmQh8M'
url = ('https://commentanalyzer.googleapis.com/v1alpha1/comments:analyze' +
    '?key=' + api_key)

# Since readability returns string of form "xth to (x+1)th grade",
# we should only grab the first one.
def find_first_num(s):
    i = re.search('[0-9]+', s).group()
    return int(i)
```

```
def features(sentence):
   d = {}
   d['readability'] = find_first_num(textstat.text_standard(sentence))
   d['length'] = len(word_tokenize(sentence))

   # preprocessing text to make readable for perspective api scores:
   text = ''
   for a in sentence:
        if a==' ' or (a<='Z' and a>='A') or (a<='z' and a>='a') or (a<='9' and a>='0') or
a=='?' or a=='.':
          text +=a

   # perspective api scores call:
       data = '{comment: {text:"'+text+'"}, languages: ["en"], requestedAttributes:
{TOXICITY:{}} }'
   response = requests.post(url=url, data=data)
   j = json.loads(response.content)
   # attempting to deal with API issues
   while 'error' in j:
      time.sleep(5)
      response = requests.post(url=url, data=data)
      j = json.loads(response.content)
   try:
      d['toxicity'] = float(j['attributeScores']['TOXICITY']['summaryScore']['value'])
   except:
      d['toxicity'] = 0.0
   assert(len(d.values()) == 3)
   return d

fileobjs[0].seek(0)
# Creating feature dict for each sample in dataset
next(readers[0], None)
all_data = list(readers[0])
```

```
feature_data = {}
for row in all_data:
    feature_data[row[0]] = features(row[1])
fileobjs[0].seek(0)

import numpy
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

def data_process(num_features):
    # Creating matrix of (samples, features) for sklearn models
    feature_matrix = []
    for i in range(1,1001):
        feature_matrix.append(list(feature_data[str(i)].values()))
    feature_matrix = numpy.array([numpy.array(x) for x in feature_matrix])
    for i in feature_matrix:
        if len(i) != num_features:
            print(i) # debugging in case perspective api fails
    return numpy.stack(feature_matrix, axis=0)

for i in fileobjs:
    i.seek(0)

feature_matrix = data_process(3)

L_classifiers_NS = []
L_classifiers_NNS = []
L_tests_NS = []
L_tests_NNS = []
for i in readers:
    next(i, None)
    list_data = list(i)
    labels_NS = [row[2] for row in list_data]
```

```
labels_NNS = [row[3] for row in list_data]

# Easier to use DataFrame obj to work with skl models
data_NS=pd.DataFrame({
    'readability':feature_matrix[:,0],
    'length':feature_matrix[:,1],
    'toxicity':feature_matrix[:,2],
    'politeness': numpy.array(labels_NS)
})
data_NS.head()
data_NNS=pd.DataFrame({
    'politeness': numpy.array(labels_NNS)
})
data_NNS.head()
X=data_NS[['readability', 'length', 'toxicity']]

# NS training
# Splitting up into 90% training, 10% verification
NS_xtrain, NS_xtest, NS_ytrain, NS_ytest = train_test_split(X, data_NS['politeness'], test_size=0.1)
L_tests_NS.append((NS_xtest, NS_ytest))

# NNS training
NNS_xtrain, NNS_xtest, NNS_ytrain, NNS_ytest = train_test_split(X, data_NNS['politeness'], test_size=0.1)
L_tests_NNS.append((NNS_xtest, NNS_ytest))

clfNS = RandomForestClassifier(n_estimators=100, max_depth=2,random_state=0)
clfNS.fit(NS_xtrain, NS_ytrain)
clfNNS = RandomForestClassifier(n_estimators=100, max_depth=2,random_state=0)
clfNNS.fit(NNS_xtrain, NNS_ytrain)
L_classifiers_NS.append(clfNS)
```

```
    L_classifiers_NNS.append(clfNNS)

for i in range(len(filenames)):
    print(filenames[i])
    print("native speaker:")
    print(L_classifiers_NS[i].score(L_tests_NS[i][0], L_tests_NS[i][1]))
    print("non-native speaker:")
    print(L_classifiers_NNS[i].score(L_tests_NNS[i][0], L_tests_NNS[i][1]))
```

parsing.py

```
# -*- coding: utf-8 -*-



import csv

# Gathering quartiles for normalized scores
csv_file_stats = open("RatingData - Sheet1.csv", "r")
csv_reader_stats = csv.reader(csv_file_stats)

scores_ns = []
scores_nns = []
next(csv_reader_stats, None)
for row in csv_reader_stats:
    scores_ns.append(float(row[4]))
    scores_nns.append(float(row[8]))
csv_file_stats.close()
scores_ns.sort()
scores_nns.sort()

partitions_ns = []
print("Quintiles for native speaker ratings:")
for i in range(6):
    p = scores_ns[int(i * (len(scores_ns) - 1)/5)]
    partitions_ns.append(p)
```

```
    print(p)

partitions_nns = []
print("Quintiles for non-native speaker ratings:")
for i in range(6):
    p = scores_nns[int(i * (len(scores_nns) - 1)/5)]
    partitions_nns.append(p)
    print(p)

import matplotlib.pyplot as plt

print("Scores by native speakers")
plt.hist(scores_ns, 20)
plt.show()
print("Scores by non-native speakers")
plt.hist(scores_nns, 20)
plt.show()

# Helper function for labeling, specs defined by labeling schemes
def getLabel(index, value, is_ns):
    if index == 0:
        # Binary labeling
        return 0 if value < 0 else 1
    elif index == 1:
        # Strong Neutral
        return 0 if abs(value) <= 0.25 else (-1 if value < 0 else 1)
    elif index == 2:
        # Weak Neutral
        return 0 if abs(value) <= 0.75 else (-1 if value < 0 else 1)
    elif index == 3:
        # Labeling with Intermediates
        if value <= -1.5:
            return -2
        elif value >= 1.5:
```

```
        return 2
      return 0 if abs(value) <= 0.5 else (-1 if value < 0 else 1)
  else:
      # Labeling with Partitions
      partitions = partitions_ns if is_ns else partitions_nns
      if value <= partitions[1]:
        return -2
      elif value >= partitions[4]:
        return 2
      elif value <= partitions[2]:
        return -1
      elif value >= partitions[3]:
        return 1
      return 0

csv_file = open("RatingData - Sheet1.csv", "r")
csv_reader = csv.reader(csv_file)

labels = ['ID', 'Message', 'NS', 'NNS']
filenames = ["BinaryLabeling.csv", "StrongNeutralLabeling.csv",
             "WeakNeutralLabeling.csv", "IntermediateLabeling.csv",
             "PartitionsLabeling.csv"]
fileobjs = [open("LabeledData/" + i, "w", newline='') for i in filenames]
writers = [csv.writer(i) for i in fileobjs]

# Gather statistics for each labeling scheme
counts_ns = [{} for i in filenames]
counts_nns = [{} for i in filenames]



for i in writers:
  i.writerow(labels)

bad_rows = 0
```

```
next(csv_reader, None)
for row in csv_reader:
    # Check for errors in comma division in csv
    if len(row) != 10:
        bad_rows += 1
    else:
        # Grabbing normalized scores from csv
        NS_score = float(row[4])
        NNS_score = float(row[8])

        # Performing labeling
        for i in range(len(filenames)):
            ns = getLabel(i, NS_score, True)
            nns = getLabel(i, NNS_score, False)
            writers[i].writerow([row[0], row[1], ns, nns])
            if ns in counts_ns[i]:
                counts_ns[i][ns] += 1
            else:
                counts_ns[i][ns] = 1
            if nns in counts_nns[i]:
                counts_nns[i][nns] += 1
            else:
                counts_nns[i][nns] = 1
csv_file.close()
for i in fileobjs:
    i.close()
print("Error rows:")
print(bad_rows)
print("\n\n")
for i in range(len(counts_ns)):
    print(filenames[i])
    print("Native speaker score frequencies:")
    print(counts_ns[i])
    print("Non-native speaker score frequencies:")
```

```
    print(counts_nns[i])
    print("\n")
```

tagger-generator.py

```
# -*- coding: utf-8 -*-


import json
import spacy
from collections import defaultdict

from convokit import Corpus, Utterance, Speaker
from convokit import download
from convokit import PolitenessStrategies, TextParser

# you will need to update the path of the downloaded/trained model in settings.py first
from strategy_manipulation import remove_strategies_from_utt,
add_strategies_to_utt

# we use spacy to obtain parses for text, which the strategy extractor rely on
spacy_nlp = spacy.load('en_core_web_sm', disable=['ner'])

# we use politeness strategy collection "politeness_local",
# i.e., a subset of strategies that can be achieved through localized markers
ps = PolitenessStrategies(strategy_attribute_name="strategies", \
                          marker_attribute_name="markers", \
                          strategy_collection="politeness_local")

message = "Is this page really still a stub?  Seems like enough information to remove
the stub marker."

strategy_plan = {"Subjunctive", "For.Me", "Gratitude"}
```

```
def edit_utterance_by_plan(message, strategy_plan, spacy_nlp, politeness_transformer):

    # importantly, we need to have markers set to be true to know the exact positions of the markers for later edits
    utt = politeness_transformer.transform_utterance(message, markers=True)

    # strategies currently used
    strategy_set = {k for k,v in utt.meta['strategies'].items() if v == 1}

    utt.meta['strategy_set'] = strategy_plan

    # We can then determine strategies that needs to be deleted, as well as strategies that should be added, by comparing strategy_plan and strategy_set.
    to_delete = strategy_set - strategy_plan
    to_add = strategy_plan - strategy_set

    remove_strategies_from_utt(utt, to_delete, removed_attribute_name='context')

    return add_strategies_to_utt(utt, to_add, politeness_transformer, spacy_nlp)

edit_utterance_by_plan(message, strategy_plan, spacy_nlp, ps)

for intended_politeness in range(-1, 3):

    print("Intended politeness level = {}".format(intended_politeness))


# load average perception model
from settings import PERCEPTION_MODEL_PATH
from plan_with_ilp import get_ilp_solution
with open(PERCEPTION_MODEL_PATH, 'r') as f:
    AVERAGE_MODEL = json.load(f)
```

```
def paraphrase_utterance_by_intention(message, intended_politeness, spacy_nlp,
politeness_transformer, perception_model = AVERAGE_MODEL):

    # importantly, we need to have markers set to be true to know the exact positions of
the markers for later edits
    utt = politeness_transformer.transform_utterance(message, markers=True)

    # strategies currently used
    strategy_set = {k for k,v in utt.meta['strategies'].items() if v == 1}

    utt.meta['strategy_set'] = strategy_set
    utt.meta['intended_politeness'] = intended_politeness
            strategy_plan = get_ilp_solution('0', strategy_set, perception_model,
perception_model, set(), intended_politeness=intended_politeness)
    print('Recommended strategy plan:', strategy_plan)

    to_delete = strategy_set - strategy_plan
    to_add = strategy_plan - strategy_set

    remove_strategies_from_utt(utt, to_delete, removed_attribute_name='context')

    return add_strategies_to_utt(utt, to_add, politeness_transformer, spacy_nlp)


for intended_politeness in range(-1, 3):

    print("Intended politeness level = {}".format(intended_politeness))

     print(paraphrase_utterance_by_intention(message, intended_politeness, spacy_nlp,
ps))
    print()
```

bleu.py

```
# script is from
https://github.com/agaralabs/transformer-drg-style-transfer/blob/master/evaluation_scripts/bleu.py
# as follow their practice

# BLEU functions from https://github.com/MaximumEntropy/Seq2Seq-PyTorch

import numpy as np
from collections import Counter
import math

def bleu_stats(hypothesis, reference):
    """Compute statistics for BLEU."""
    stats = []
    stats.append(len(hypothesis))
    stats.append(len(reference))
    for n in range(1, 5):
        s_ngrams = Counter(
            [tuple(hypothesis[i:i + n]) for i in range(len(hypothesis) + 1 - n)]
        )
        r_ngrams = Counter(
            [tuple(reference[i:i + n]) for i in range(len(reference) + 1 - n)]
        )
        stats.append(max([sum((s_ngrams & r_ngrams).values()), 0]))
        stats.append(max([len(hypothesis) + 1 - n, 0]))
    return stats

def bleu(stats):
    """Compute BLEU given n-gram statistics."""
    if len(list(filter(lambda x: x == 0, stats))) > 0:
        return 0
    (c, r) = stats[:2]
```

```
    log_bleu_prec = sum(
        [math.log(float(x) / y) for x, y in zip(stats[2::2], stats[3::2])]
    ) / 4.
    return math.exp(min([0, 1 - float(r) / c]) + log_bleu_prec)

def get_bleu(hypotheses, reference):
    """Get validation BLEU score for dev set."""
    stats = np.array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
    for hyp, ref in zip(hypotheses, reference):
        stats += np.array(bleu_stats(hyp, ref))
    return 100 * bleu(stats)
```

be-af.py

```
# -*- coding: utf-8 -*-


import pandas as pd
import os

from convokit import Corpus, Utterance, Speaker
from convokit import PolitenessStrategies
train_corpus = Corpus(filename=("data/train/training-corpus/"))
ps = PolitenessStrategies(strategy_attribute_name = "strategies", \
                          marker_attribute_name = "markers", \
                          strategy_collection="politeness_local")
# it is important to set markers to True
train_corpus = ps.transform(train_corpus, markers=True)

for utt in train_corpus.iter_utterances():

    strategy_split = utt.meta['strategy']
    assert utt.meta['strategies'][strategy_split] == 1
```

```
# helper functions further detailed in Marker_Edits.ipynb
from strategy_manipulation import remove_strategies_from_utt

for utt in train_corpus.iter_utterances():

    remove_strategies_from_utt(utt, [utt.meta['strategy']])

utt = train_corpus.get_utterance('100087711.41.31')
#           100087711.41.31
#           10387534.0.0
#           105319599.26773.0

print("BEFORE:", utt.text)
print("AFTER:", utt.meta)
```

Evaluation_data.py

```
# -*- coding: utf-8 -*-

import pandas as pd
import numpy as np
import json
import os
import spacy
from collections import Counter
from tqdm import tqdm

import json
import spacy
from collections import defaultdict

from convokit import Corpus, Utterance, Speaker
```

```
from convokit import download
from convokit import PolitenessStrategies, TextParser

# you will need to update the path of the downloaded/trained model in settings.py first
from strategy_manipulation import remove_strategies_from_utt,
add_strategies_to_utt

mt_corpus = Corpus(filename="data/test/mt-test-corpus")

mt_corpus.get_utterance('226408211.416.416').meta

ps = PolitenessStrategies(strategy_attribute_name = "strategies", \
                    marker_attribute_name = "markers", \
                    strategy_collection="politeness_local")

spacy_nlp = spacy.load('en', disable=['ner'])

# get politeness strategies for the back translations
translated_strategies = {utt.id: ps.transform_utterance(utt.meta['back_translation'], \
                                        spacy_nlp=spacy_nlp, markers=True) for utt in
mt_corpus.iter_utterances()}



cnts = {'Subjunctive': 0, "Please": 0, 'Filler': 0, "Swearing": 0}

for utt in mt_corpus.iter_utterances():

    k = utt.meta['strategy']
    cnts[k] += translated_strategies[utt.id].meta['strategies'][k]

print("===Strategy Preservation Rate===")
for k, v in cnts.items():

    # counts are out of a total of 200 instances
```

```
    print("{}: {:.1f}% ".format(k.upper(), v/2))

ind_corpus = Corpus(filename="data/test/ind-test-corpus/")

ind_corpus.get_utterance("A23-A3S-1").meta

pairs = [(utt.meta['sender'], utt.meta['receiver']) for utt in ind_corpus.iter_utterances()]

from collections import defaultdict
from perception_utils import scale_func, get_strategies_df, get_model_info,
get_ind_model_info

wiki_politeness = Corpus(download("wikipedia-politeness-corpus"))

wiki_politeness = ps.transform(wiki_politeness, markers = False)

for utt in wiki_politeness.iter_utterances():

    utt.meta['avg_score'] = scale_func(np.mean(list(utt.meta['Annotations'].values())))

df_avg = get_strategies_df(wiki_politeness, 'strategies')
scores = [wiki_politeness.get_utterance(idx).meta['avg_score'] for idx in
df_avg.index]

avg_model = get_model_info(df_avg, scores)
avg_model

turker_corpus = Corpus(filename="data/perceptions/turker-corpus/")

Counter([utt.speaker.id for utt in turker_corpus.iter_utterances()])

turker_corpus = ps.transform(turker_corpus, markers=True)

turkers = ['A23', 'A2U', 'A1F', 'A3S', 'AYG']
```

```
turker_dfs = defaultdict()

for turker in turkers:

    corpus = turker_corpus.filter_utterances_by(selector=lambda utt:utt.speaker.id == turker)

    df = get_strategies_df(corpus, "strategies")

    df['score'] = [corpus.get_utterance(idx).meta['score'] for idx in df.index]

    turker_dfs[turker] = df

avg_coefs = avg_model['coefs']

for t, df in turker_dfs.items():

    df_feat = df.iloc[:, 0:-1]
    scores = dict(df.iloc[:, -1])

    ind_model = get_ind_model_info(df_feat, scores, avg_coefs)

    print(t)
    print(ind_model)
    print('======')
```

4.3 TEST CASES

```
Better speed can be achieved with apex installed from https://www.github.com/nvidia/apex.
Note: using BERT BasicTokenizer instead of SpaCy.
BEFORE: Surely this isn't "of interest" to a robotics project? '' '', .
```

```
AFTER: {'strategy': 'Actually', 'parsed': [{'rt': 2, 'toks': [{'tok': 'surely', 'tag': 'RB', 'dep':
'advmod', 'up': 2, 'dn': []}, {'tok': 'this', 'tag': 'DT', 'dep': 'nsubj', 'up': 2, 'dn': []},
{'tok': 'is', 'tag': 'VBZ', 'dep': 'ROOT', 'dn': [0, 1, 3, 4, 5, 8, 12, 13, 15, 16, 17]}, {'tok':
"n't", 'tag': 'RB', 'dep': 'neg', 'up': 2, 'dn': []}, {'tok': '"', 'tag': '``', 'dep': 'punct',
'up': 2, 'dn': []}, {'tok': 'of', 'tag': 'IN', 'dep': 'prep', 'up': 2, 'dn': [6, 7]}, {'tok':
'interest', 'tag': 'NN', 'dep': 'pobj', 'up': 5, 'dn': []}, {'tok': '"', 'tag': "''", 'dep':
'punct', 'up': 5, 'dn': []}, {'tok': 'to', 'tag': 'IN', 'dep': 'prep', 'up': 2, 'dn': [11]},
{'tok': 'a', 'tag': 'DT', 'dep': 'det', 'up': 11, 'dn': []}, {'tok': 'robotics', 'tag': 'NNS',
'dep': 'compound', 'up': 11, 'dn': []}, {'tok': 'project', 'tag': 'NN', 'dep': 'pobj', 'up': 8,
'dn': [9, 10]}, {'tok': '?', 'tag': '.', 'dep': 'punct', 'up': 2, 'dn': []}, {'tok': "''", 'tag':
"''", 'dep': 'punct', 'up': 2, 'dn': [14]}, {'tok': '\xa0', 'tag': '_SP', 'dep': '', 'up': 13,
'dn': []}, {'tok': "''", 'tag': "''", 'dep': 'punct', 'up': 2, 'dn': []}, {'tok': ',', 'tag': ',',
'dep': 'punct', 'up': 2, 'dn': []}, {'tok': '.', 'tag': '.', 'dep': 'punct', 'up': 2, 'dn': []}]}],
'split': 'train', 'strategies': {'By.The.Way': 0, 'Filler': 0, 'Greeting': 0, 'Hedges': 0,
'Indicative': 0, 'Subjunctive': 0, 'For.Me': 0, 'For.You': 0, 'Reassurance': 0, 'Conj.Start': 0,
'Affirmation': 0, 'Please.Start': 0, 'Actually': 1, 'Adverb.Just': 0, 'Apology': 0, 'Gratitude': 0,
'Please': 0, 'Swearing': 0}, 'markers': {'By.The.Way': [], 'Filler': [], 'Greeting': [], 'Hedges':
[], 'Indicative': [], 'Subjunctive': [], 'For.Me': [], 'For.You': [], 'Reassurance': [],
'Conj.Start': [], 'Affirmation': [], 'Please.Start': [], 'Actually': [('surely', 0, 0)],
'Adverb.Just': [], 'Apology': [], 'Gratitude': [], 'Please': [], 'Swearing': []},
'post_del_context': 'this is n\'t " of interest " to a robotics project ? \'\' \xa0 \'\' , .'}
```


Fig 4.3.1: Test case 1

```
Note: using BERT BasicTokenizer instead of SpaCy.
BEFORE: Is this page really still a stub?  Seems like enough information to remove the stub marker.
```

```
AFTER: {'strategy': 'Actually', 'parsed': [{'rt': 0, 'toks': [{'tok': 'is', 'tag': 'VBZ', 'dep':
'ROOT', 'dn': [2, 4, 6, 7, 9, 18]}, {'tok': 'this', 'tag': 'DT', 'dep': 'det', 'up': 2, 'dn': []},
{'tok': 'page', 'tag': 'NN', 'dep': 'nsubj', 'up': 0, 'dn': [1]}, {'tok': 'really', 'tag': 'RB',
'dep': 'advmod', 'up': 4, 'dn': []}, {'tok': 'still', 'tag': 'RB', 'dep': 'advmod', 'up': 0, 'dn':
[3]}, {'tok': 'a', 'tag': 'DT', 'dep': 'det', 'up': 6, 'dn': []}, {'tok': 'stub', 'tag': 'NN',
'dep': 'attr', 'up': 0, 'dn': [5]}, {'tok': '?', 'tag': '.', 'dep': 'punct', 'up': 0, 'dn': [8]},
{'tok': ' ', 'tag': '_SP', 'dep': '', 'up': 7, 'dn': []}, {'tok': 'seems', 'tag': 'VBZ', 'dep':
'npadvmod', 'up': 0, 'dn': [10]}, {'tok': 'like', 'tag': 'IN', 'dep': 'prep', 'up': 9, 'dn': [12]},
{'tok': 'enough', 'tag': 'JJ', 'dep': 'amod', 'up': 12, 'dn': []}, {'tok': 'information', 'tag':
'NN', 'dep': 'pobj', 'up': 10, 'dn': [11, 14]}, {'tok': 'to', 'tag': 'TO', 'dep': 'aux', 'up': 14,
'dn': []}, {'tok': 'remove', 'tag': 'VB', 'dep': 'acl', 'up': 12, 'dn': [13, 17]}, {'tok': 'the',
'tag': 'DT', 'dep': 'det', 'up': 17, 'dn': []}, {'tok': 'stub', 'tag': 'NN', 'dep': 'compound',
'up': 17, 'dn': []}, {'tok': 'marker', 'tag': 'NN', 'dep': 'dobj', 'up': 14, 'dn': [15, 16]},
{'tok': '.', 'tag': '.', 'dep': 'punct', 'up': 0, 'dn': []}]}], 'split': 'train', 'strategies':
{'By.The.Way': 0, 'Filler': 0, 'Greeting': 0, 'Hedges': 0, 'Indicative': 0, 'Subjunctive': 0,
'For.Me': 0, 'For.You': 0, 'Reassurance': 0, 'Conj.Start': 0, 'Affirmation': 0, 'Please.Start': 0,
'Actually': 1, 'Adverb.Just': 0, 'Apology': 0, 'Gratitude': 0, 'Please': 0, 'Swearing': 0},
'markers': {'By.The.Way': [], 'Filler': [], 'Greeting': [], 'Hedges': [], 'Indicative': [],
'Subjunctive': [], 'For.Me': [], 'For.You': [], 'Reassurance': [], 'Conj.Start': [], 'Affirmation':
[], 'Please.Start': [], 'Actually': [('really', 0, 3)], 'Adverb.Just': [], 'Apology': [],
'Gratitude': [], 'Please': [], 'Swearing': []}, 'post_del_context': 'is this page still a stub ?
seems like enough information to remove the stub marker .'}
```


Fig 4.3.2: Test case2

```
pytorch_pretrained_bert, generation_utils, perception_utils, settings, strategy_manipulation
Better speed can be achieved with apex installed from https://www.github.com/nvidia/apex.
Note: using BERT BasicTokenizer instead of SpaCy.
BEFORE: Is this page really still a stub?  Seems like enough information to remove the stub marker.
AFTER: {'strategy': 'Actually', 'parsed': [{'rt': 0, 'toks': [{'tok': 'is', 'tag': 'VBZ', 'dep':
'ROOT', 'dn': [2, 4, 6, 7, 9, 18]}, {'tok': 'this', 'tag': 'DT', 'dep': 'det', 'up': 2, 'dn': []},
{'tok': 'page', 'tag': 'NN', 'dep': 'nsubj', 'up': 0, 'dn': [1]}, {'tok': 'really', 'tag': 'RB',
'dep': 'advmod', 'up': 4, 'dn': []}, {'tok': 'still', 'tag': 'RB', 'dep': 'advmod', 'up': 0, 'dn':
[3]}, {'tok': 'a', 'tag': 'DT', 'dep': 'det', 'up': 6, 'dn': []}, {'tok': 'stub', 'tag': 'NN',
'dep': 'attr', 'up': 0, 'dn': [5]}, {'tok': '?', 'tag': '.', 'dep': 'punct', 'up': 0, 'dn': [8]},
{'tok': ' ', 'tag': '_SP', 'dep': '', 'up': 7, 'dn': []}, {'tok': 'seems', 'tag': 'VBZ', 'dep':
'npadvmod', 'up': 0, 'dn': [10]}, {'tok': 'like', 'tag': 'IN', 'dep': 'prep', 'up': 9, 'dn': [12]},
{'tok': 'enough', 'tag': 'JJ', 'dep': 'amod', 'up': 12, 'dn': []}, {'tok': 'information', 'tag':
'NN', 'dep': 'pobj', 'up': 10, 'dn': [11, 14]}, {'tok': 'to', 'tag': 'TO', 'dep': 'aux', 'up': 14,
'dn': []}, {'tok': 'remove', 'tag': 'VB', 'dep': 'acl', 'up': 12, 'dn': [13, 17]}, {'tok': 'the',
'tag': 'DT', 'dep': 'det', 'up': 17, 'dn': []}, {'tok': 'stub', 'tag': 'NN', 'dep': 'compound',
'up': 17, 'dn': []}, {'tok': 'marker', 'tag': 'NN', 'dep': 'dobj', 'up': 14, 'dn': [15, 16]},
{'tok': '.', 'tag': '.', 'dep': 'punct', 'up': 0, 'dn': []}]}], 'split': 'train', 'strategies':
{'By.The.Way': 0, 'Filler': 0, 'Greeting': 0, 'Hedges': 0, 'Indicative': 0, 'Subjunctive': 0,
'For.Me': 0, 'For.You': 0, 'Reassurance': 0, 'Conj.Start': 0, 'Affirmation': 0, 'Please.Start': 0,
'Actually': 1, 'Adverb.Just': 0, 'Apology': 0, 'Gratitude': 0, 'Please': 0, 'Swearing': 0},
'markers': {'By.The.Way': [], 'Filler': [], 'Greeting': [], 'Hedges': [], 'Indicative': [],
'Subjunctive': [], 'For.Me': [], 'For.You': [], 'Reassurance': [], 'Conj.Start': [], 'Affirmation':
[], 'Please.Start': [], 'Actually': [('really', 0, 3)], 'Adverb.Just': [], 'Apology': [],
'Gratitude': [], 'Please': [], 'Swearing': []}, 'post_del_context': 'is this page still a stub ?
seems like enough information to remove the stub marker .'}
```

Fig 4.3.3: Test case 3

4.4 ACCURACY

```
Fold  9
Epoch 1/40
52/52 [==============================] - 28s 390ms/step - loss: 0.5240 - accuracy: 0.8102 -
val_loss: 0.4445 - val_accuracy: 0.8019
Epoch 2/40
52/52 [==============================] - 21s 406ms/step - loss: 0.3452 - accuracy: 0.8326 -
val_loss: 0.4180 - val_accuracy: 0.8599
Epoch 3/40
52/52 [==============================] - 468s 9s/step - loss: 0.1492 - accuracy: 0.9421 - val_loss:
0.5991 - val_accuracy: 0.8502
Epoch 4/40
52/52 [==============================] - 15s 279ms/step - loss: 0.0451 - accuracy: 0.9830 -
val_loss: 0.8403 - val_accuracy: 0.8454
Epoch 5/40
52/52 [==============================] - 16s 308ms/step - loss: 0.0300 - accuracy: 0.9940 -
val_loss: 0.6718 - val_accuracy: 0.8213
Train: 1.000, Test: 0.821
              precision    recall  f1-score   support

           0       0.94      0.94      0.94       180
           1       0.59      0.62      0.60        26

    accuracy                           0.90       206
   macro avg       0.77      0.78      0.77       206
weighted avg       0.90      0.90      0.90       206
```

Fig 4.4: Accuracy

4.5 POLITENESS SCORES

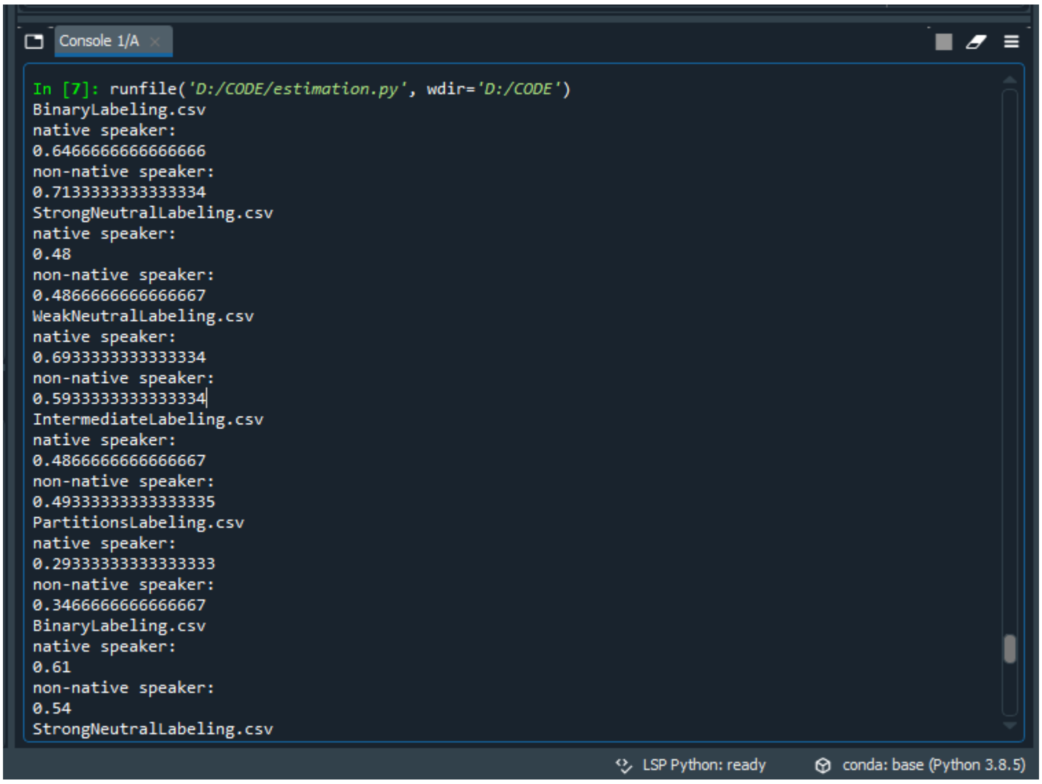


Fig 4.4.1: Politeness Score

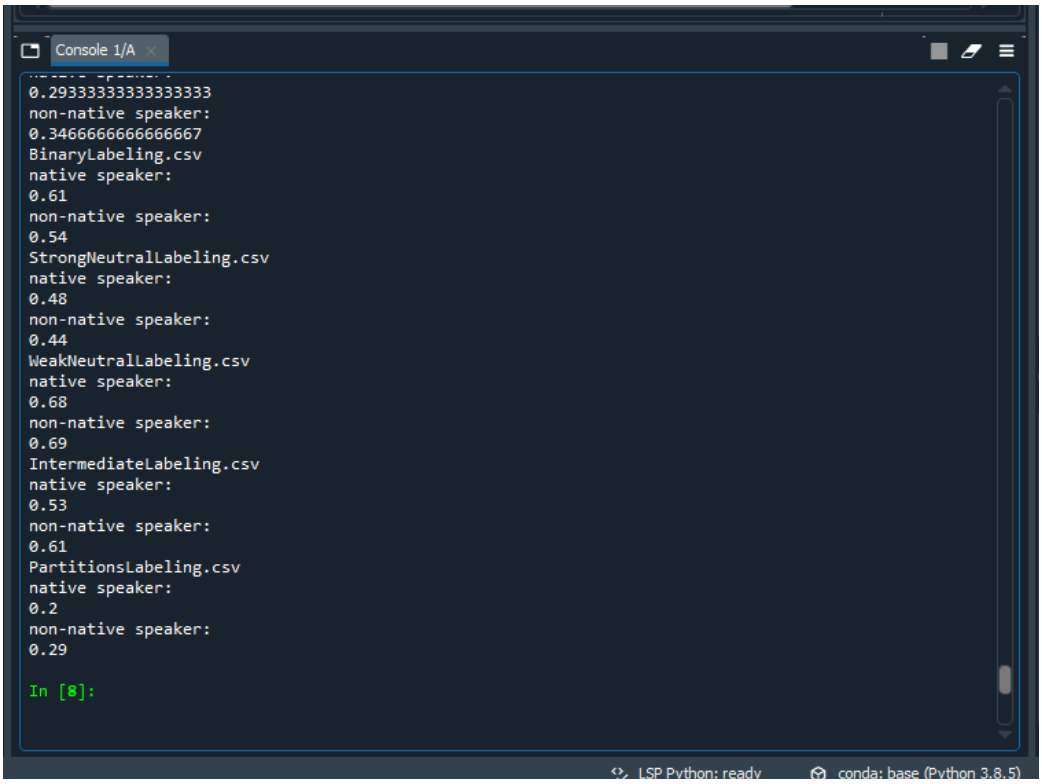


Fig 4.4.2: Politeness Score

5. CONCLUSION

We introduce the task of politeness transfer for which we provide a dataset composed of sentences accurate from email exchanges present in the Enron corpus. We extend prior works on attribute transfer by introducing a simple pipeline – tag & generate which is an interpretable two-staged approach for content preserving style transfer. We believe our approach is the first to be robust in cases when the source is style neutral, like the "non-polite" class in the case of politeness transfer.

6. FUTURE SCOPE

Automatic and human evaluation shows that our approach outperforms other state-of-the-art models on content preservation metrics while retaining (or in some cases improving) the transfer accuracies.

6. REFERENCES

[1] Aman Madaan, Amrith Setlur, Tanmay Parekh , Barnabas Poczos, Graham Neubig, Yiming Yang, Ruslan Salakhutdinov, Alan W Black, Shrimai Prabhumoye. (2019) '*Politeness Transfer: A Tag and Generate Approach*' School of Computer Science Carnegie Mellon University Pittsburgh, PA, USA. (Base paper).

[2] Penelope Brown, Stephen C Levinson, and Stephen C Levinson. 1987. Politeness: Some universals in language usage, volume 4. Cambridge university press.

[3] Philip Bramsen, Martha Escobar-Molano, Ami Patel, and Rafael Alonso. 2011. Extracting social power relationships from natural language. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1, HLT '11, pages 773–782, Stroudsburg, PA, USA. Association for Computational Linguistics

[4] Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 250–259, Sofia, Bulgaria. Association for Computational Linguistics.

[5] Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In the Thirty-Second AAAI Conference on Artificial Intelligence.