# Assignment - 1

## Inhabitant term prediction

---

210050166 - Vir Wankede - CSE
200050082 - Nagesh Kumar - CSE
200050052 - Hemendra Meena - CSE

September 3, 2023

# Table of contents

1. Problem Statement
2. Working with Data
    - Data Scraping
    - Pre-Processing
    - Vocabulary Size
3. Models
    - Simple Rule Based Approach
4. Byte Pair Encoding Approach
5. Results
    - Simple Rule Model
    - BPE Rule Model
6. Analysis
7. Model Comparison

# Problem Statement

# Problem Statement

**Data collection**: Collect/Create data of city/town/village vs. inhabitant-term. Create train, validation and test splits **Input**: A city/place name: London

**Output**: Inhabitant term: Londoner You will have to report on the validation and test data:

- Accuracy
- Perform detailed error analysis

# Working with Data

# Data Scraping

- We used Wikipedia and Github to get the dataset
- From these source we got around 1500 length of dataset
- Sources:-
    - *https://en.wikipedia.org/wiki/List_of_adjectival_and_demonymic_forms_of_place_names#Indian_states_and_territories*
    - *https://github.com/knowitall/chunkedextractor/blob/master/src/main/resources/edu/knowitall/chunkedextractor/demonyms.csv*

# Pre-Processing

- **Deduplication** - Removed multiple cases when country having more than one inhabitant like for Ireland we can have demonym as Irish, Irishman.
- **Ambiguous Inhabitant Term** - We removed most of the term where inhabitant term was completely ambiguous to the city name like for Santa Catarina(a place in Brazil) have inhabitant term Barriga-Verde
- **Stemming** - Stemming is also done but rather than doing it for data we have done it inside function find_inhabitant_term to ensure any new words can also be used without separately doing pre-processing for that. This is done using **PorterStemmer()** function of NLTK library
- **Data shuffling** - Helped the data to get randomized because initially, it was in the alphabetic order.This helped the training model.
- **Data Splitting** - The data was split in the ratio **8:1:1** for training, test and validation.
- **Generating corpus** - **Inhabitant term words** are used as the

After **deduplication** and **removal of ambiguous name**, out of around **1500** terms we we left with **1100** names.

# Vocabulary Size

In case of BPE as we used training set, we use around **800 words** to train the model.

# Models

# Type of Models used

We have used two rule-based approach :-

- Simple Rule Based(without BPE)
- BPE Rule Based

# Simple Rule Based Approach

Idea behind model

- We analysed the data
- Find a pattern in inhabitant term
- Based on that implemented some rules

About Model

- The simple rule model just has a few basic rules without efficient and robust sub tokenization of the strings.
- The **suffix consisting of 1 character** was considered, and based on inference were matched with certain demonyms.

# Library Used

1. **NLTK**
   In this we used function NLTK.stem.PorterStemmer for stemming
2. **Pandas**
   For dataset operations

# Model Architecture

- First we initialise the rules in dictionary based on the last character of word like 'a' : 'an',etc.
- We made function find_inhabitant_term which takes input the name
- This function first performs stemming on the name and then check the last character of the word
- If word is in rules then it adds the suffix according to rules otherwise a default 'er' suffix is added

# Why Simple Rule model?

- While going through the data we realised that there is some pattern in with character words ends and suffix is added to make demonym
- Thus it seemed to be first approach to start working on the problem.

# Parameters

Since we used simple rule based approach whereby we made rules based on our own inference of data. So there were no parameters to change in the code.

# Byte Pair Encoding Approach

# BPE Model

For this model

- Used a training set(of inhabitant) to **generate rules** using BPE algorithm
- Later **tokenized** the test set words
- On these token, we **used rules** (made using training of BPE) to predict Inhabitant term

# Library used

1. **NLTK**
   In this we used function NLTK.stem.PorterStemmer for stemming
2. **Pandas**
   For dataset operations
3. **Subword_nmt**
   This is used for BPE algorithm and two function are used which are subword_nmt.apply_bpe(for tokenizing) and subword_nmt.learn_bpe(for training BPE)
4. **Sklearn**
   In this we used sklearn.shuffle

# Model Architecture

- Firstly we pre-process the data and distribute it in Train, Test and Validation set
- Then perform BPE on the Inhabitant terms in Training set using subword_nmt.learn_bpe function.
- This generates a file containing rule for generating which we load in using function subword_nmt.apply_bpe which is later used for tokenizing the words
- After that we load the generated rules in bpe_rules dictionary with following condition:-
    - The most repeated rule is considered
    - Only those rules taken which follows word end character
- After this we have find_inhabitant_term function which takes input a word and bpe_rules(above generated)
- This function first performs stemming on word and generates its token.

# Model Architecture (Continued)

- This function iterates over token finding then is bpe_rules from start to end.
  For eg:- If words is India and has tokens i, n, d, ia then the iteration will search following india, ndia, dia, ia until it find the rule related to this.
  This is to ensure rule with max length is find.
- Then if no word is found the character wise search on last token is made.
- On finding rules related to them we add the suffix in the last and return the word
- If still no rule found add ’er’ by default.

# Why BPE?

- BPE proved to be a good subword tokenizating model and could handle the variance in the city names.
- It allowed the code to take a rule-based approach to generate demonyms.It could find the longest matching rule for a city name by iterating through the subword units.
- The rules in this case are the mergers, for characters that occur adjacent to each other with a high frequency.

# Parameters

The BPE is a rule based approach, so not many parameters or hyper-parameters were needed which are generally used for neural networks and deep learning models.
However, there were some parameters used even in BPE, like num_operations. This parameter defined the number of which were to be performed during the training phase. In the code, it was set to 1000.

# Results

# Results - Simple Rule Based

Since **Simple Rule** model based approach does used Training set, so we used whole set to get accuracy
**Accuracy** - **36.16**

# Results - BPE Rule Based

For BPE Rule based model, there is no parameter so no need for validation data.
Follwing are accuracy on different dataset :-

- On Test set, Accuracy - 41.96
- On Validation set, Accuracy - 36.36
- On Training set, Accuracy - 43.07
- On whole data, Accuracy - 42.29

# Analysis

# Error Analysis

Error in prediction can be due to :-

- Ambiguity in the naming sense of inhabitant term like for France we have French and for Ireland we have Irish which are difficult to find using any pattern or rule based approach
- The **less amount of data** (around 1100 name) that we were able to find also lead decreased size of vocabulary and so the rules generated by BPE become less effective
- The **limitation of rule based approach** are clear due to the low accuracy. We think that a Machine Learning model will be able to give higher accuracy on this.

# Model Comparison

# Simple Rule vs BPE

- BPE generated rules on its own unlike simple rule approach where we ourselves have analyse the data to find the rules.
- Thus is also more preferable in case we have bigger dataset.
- BPE proved to be a good subword tokenizating model and could handle the variance in the city names.
- It allowed the code to take a rule-based approach to generate demonyms.It could find the longest matching rule for a city name by iterating through the subword units.
- The rules in this case are the mergers, for characters that occur adjacent to each other with a high frequency.