Indian Institute of Information Technology, Nagpur

End Semester Project

Subject: Natural Language Processing

Topic: Sentiment Analysis in Hindi using Ulmfit

Submitted To: Dr Pooja Jain

Introduction:

What is Sentiment Analysis?

Sentiment analysis is the interpretation and classification of emotions (positive, negative and neutral) within text data using text **analysis** techniques. **Sentiment analysis** allows businesses to identify customer **sentiment** toward products, brands or services in online conversations and feedback.

Usually, most of the NLP techniques are applied to widely spoken languages like **English**. There is a dearth of text processing techniques for less spoken languages like **Hindi**. Out of the various problems, here we have tried to tackle one of the fundamental problems of **Sentiment Analysis in Hindi**

Transfer Learning:

Deep learning requires a **lot of data**. And moreover the neural net gets trained for a particular task at hand. So what to do if we have less amount of data? The answer is **Transfer Learning.**

Transfer learning basically involves the **'transfer' of knowledge** from a large dataset to a smaller dataset representative of the task at hand. This is done by training the neural net first on a large corpus and then using the smaller dataset to **'fine tune'** the model for the problem.

Transfer learning techniques have been used a lot before in the field of **Computer Vision.** Here we deploy a similar concept by using the **Fast ai** library that implements those concepts using robust techniques and is able to achieve SOTA in various NLP related tasks.

UlmFit (Universal Language Model for Fine Tuning):

Ulmfit is essentially a method to enable transfer learning for any NLP task and achieve great results,without having to train models from scratch. It uses a set of novel techniques like:

- Discriminative fine-tuning
- Slanted triangular learning rates, and
- Gradual unfreezing

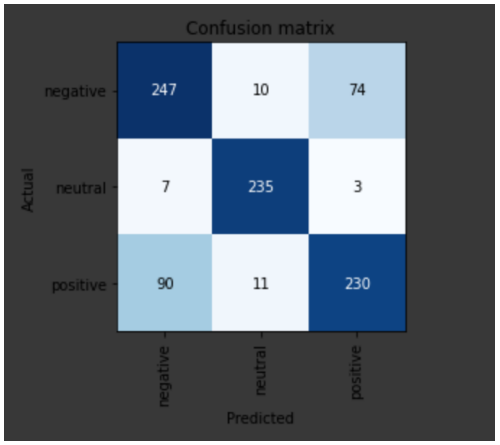
Dataset and Implementation:

- We use a large corpus of around **10,000 Hindi labelled tweets** and a mixture of **movie reviews.**
- We start by **preprocessing** the text using **SpaCy's** tokenizer and create a 'Databunch'.
- We then fine tune a **language model (AWD-LSTM)** using the preprocessed text by **gradually unfreezing** the layers of the LSTM. This ensures that the model is able to learn the nitty gritty of the dataset
- We save the '**encoder'** of our fine tuned mode as it stores the task specific information
- The encoder is further used to train a **classifier**, which predicts the sentiment of an instance. We achieved an **accuracy of 82%.**

To make sure we get better results some of the techniques used are:

- We use **mixed precision** for greater speed, smaller memory footprint, and a regularizing effect.
- We adjust the value of **dropout among layers**. A larger dropout helps prevent overfitting.
- We find the **optimal learning rate** for training by using the inbuilt function to find the value at which the loss is the least.
- We set the '**Discriminative learning rates**', so that the last layer has a smaller learning rate than the initial layers as it requires more learning.
- We make sure that the **training loss is greater than the validation loss,** to avoid overfitting.
- We even try an "**Ensemble Approach**", wherein we train a backwards language model, by feeding it text in reverse. Then we run both the models and average out the predictions. We obtain an **accuracy of 78%**

Confusion Matrix



Training and Validation loss

epoch	train_loss	valid_loss	accuracy	time
0	0.499995	0.426364	0.805954	00:20
1	0.494149	0.421116	0.811466	00:22
2	0.470588	0.414140	0.820287	00:21
3	0.487782	0.414364	0.812569	00:19
4	0.462538	0.407951	0.822492	00:20

Deployment:

To create a product we have deployed our model locally by providing an intuitive User Interface which, given an input, predicts the sentiment of the instance. We also provide an option for users to **expand our databas**e and improve our model by providing the facility of annotating data.

Tech stack (Client Side): JavaScript (NodeJS)

Tech stack (Server Side): Python (Flask)

References:

1) Text Classification (NLP) Tutorial In Python
2) fast.ai NLP · Practical NLP
3) Understanding language modelling(NLP) and Using ULMFIT
4) NLP (Sentiment Analysis) — Hindi!!! - Siddhant Sinha

Submitted By:
Gaurav Agarwal (BT17CSE001)
Abhibha Gupta (BT17CSE020)