Submission by

Anup Bhutada

[anup.bhutada44@gmail.com](mailto:anup.bhutada44@gmail.com)

+91 9545765487

I have proceeded to solve the problem with the assumption that only text data in “pavbhaji.json” is to be used for training classification models and no images need to be used in the “dataset” folder.

I started by exploring the text data and trying to find what kind of patterns exist in the posts of images with and without pavbhaji. Turns out, a lot of the words like “pavbhaji”, “yummy”, “foodie” that appear in pavbhaji posts also appear in non-pavbhaji posts making it difficult for classification tasks. I tried to check if there were differences in word occurrences when hashtags were ignored. I analysed this on the term “pavbhaji” since this word would make the most impact. I found that on ignoring hashtags, there remained very few posts with the mention of the word “pavbhaji”. The results of this analysis are shown in the notebook “exploratory\_analysis.ipynb”.

Since the distribution of words does not directly appear to be showing any useful patterns, I tried to check whether some order exists in their positions. On inspection, I found that within the data about 90% of the posts in pavbhaji class have “pavbhaji” in the first 40% of the text content. But to my disappointment, the non-pavbhaji class did not exhibit any such a pattern. It had random positions for the term “pavbhaji” in its posts.

Thereafter, I went on to inspect the images of the classes and noticed that some images that clearly contained pavbhaji in them were present in the non-pavbhaji class directory. So I went through all the images and tried to transfer images with a single pavbhaji in them to the “1” directory from “0” directory.

With this analysis, I proceeded to try out the following methods to classify the text documents.

1. Pretrained sentence encoders
2. Bag of words
3. Bag of words with positions
4. LSTM on Code-mixing embeddings of hindi and english
5. Topic modelling

Since I could not find any notable patterns in the exploratory analysis, I started out with the powerful sentence encoder models, in hopes of discovering some pattern to help in developing a more analytical model. I preprocessed the text to remove @mentions and punctuations including ‘#’ symbol. I also replaced all possible variations of the term “pavbhaji” (pav bhaji, #pavbhaji, Pav Bhaji) with the term “pavbhaji”. I then tried out tensorflow hub’s universal sentence encoder and NNLM model with customized top layers to predict the output class. These experiments are shown in the notebook “classification\_using\_sentence\_encoders.ipynb”. I did not achieve great results and the best accuracy was around 75% on the validation set.

So then I proceeded to the simpler bag of words model. I used ngrams of upto 5 tokens and achieved an accuracy of 73% to 74% on the validation set. To exploit the positional pattern that I had noticed earlier, I tried an approach wherein instead of the counts/frequency of the words/ngrams, I populated the document vectors with the position of the first occurrence of the word/ngram in the document. This approach achieved an accuracy of almost 75%, not very far from the bag of words approach. Both these experiments are shown in the notebook “bow\_model\_classification.ipynb”.

To achieve higher accuracies, it was clear that the context in which the term “pavbhaji” appeared in the documents needed to be accounted for. For example, if pavbhaji was the central subject of the post, it was more likely to be a picture of pavbhaji and vice versa. I tried to use an LSTM RNN model for this purpose with the pretrained ULMFiT embeddings from inltk. These embedding had a perplexity of 86.48 on the hindi-english code mixed language (not an encouraging fact). In the experiment with a bidirectional LSTM, I was able to achieve an accuracy of 71% on the validation set. The experiment is shown in “Classification\_using\_codemixing\_embeddings.ipynb”.

Finally, I tried topic modelling to predict topic weights in each document and use these weights as features for a classifier. This approach yielded an accuracy of 62% and is demonstrated in “lda\_on\_text.ipynb”.

I had a few more ideas that I wanted to try out.

One was to pick a smaller text window from each document containing the word “pavbhaji” or its variations so that the model would pay more attention to its context.

Another was to extract features from the main body of the post and the hashtags separately and then pass it to a classifier to predict output class.

Thank you for the opportunity.