Data collection process

Training is the most important part of any machine learning model. It is through the data that we use to train the model is how machine's accuracy changes. We always need to ensure that the kind of data we choose is not specific to a particular attribute. For example, If you have trained your model only on sentences that are 20 words long, this wouldn't work well in the testing phase, because in the real-world the sentences can be of any arbitrary length.

For my project, I have ensured to collect different types of data from several known sources. The data I choose for the Italian sentences is from a Kaggle dataset of old newspapers, however, it was a dataset of over 100,000 sentences arranged in an alphabetical order. To pick 2,500 different sentences from this dataset I have used the following python code:

```
def create_train_data():
    input = open("RawIt.txt", encoding="utf8")
    output = open("train", 'a', encoding="utf8")
    lines = 0
    current = 0
    for each in input:
        if each != "\n" and current%50 == 0:
            output.write("it| " + each )
            lines += 1
        current += 1
        if lines == 2500:
            break
    input.close()
```

This helped me collect random sentences with different features. While storing the sentences in my training data set I have ensured to add "it|" label in the beginning of every sentence indicating that it is written in Italian.

To collect sentences in Dutch I have used the Wikipedia articles, I first found a list of all Wikipedia Dutch articles, put them in a list and then passed them into the function written in the python file snippet added below. Python's Wikipedia api

tools helped me collect the data easily. All these raw sentences were then put into a text file, after which I processed the text file.

- I removed all the unnecessary characters in sentences, python's regex function came handy.
- I wrote a python code to remove all the trailing whitespaces.
- The same python code helped me remove extra white lines

By the end of this phase, I had a clean, compact and a processed data. Now I used this data, and appended it to the main train dataset with a label "nl|" appended at the beginning of every sentence, indicating it is written in Dutch.

```
import wikipediaapi

def create_rawDutch(pages):
    for eachPage in pages:
        wiki_wiki =
wikipediaapi.Wikipedia(language='nl',
extract_format=wikipediaapi.ExtractFormat.WIKI)
        p_wiki = wiki_wiki.page(eachPage)
        dutchFile = open("dutch", 'a', encoding="utf8")
        dutchFile.write(p_wiki.text)
```

In the end, I had a good train dataset with 2,500 Dutch sentences and 2,500 Italian sentences.

I collected another set of Italian and Dutch sentences similarly and appended 500 Italian and 500 Dutch sentences to the test data set and another set of 500 Italian and 500 Dutch sentences to Validation data set. 2500 train data: 1000 test data, the train data to test data ratio was 5:2

After collecting and processing the data, the next big step was to find features. Based on some research and some data exploration techniques, I found a total of 6 features for my model.

- Has Apostrophe: Most of the Italian words have a lot of special characters and some special alphabets, like accent. Most common one is an apostrophe.

Most of the Dutch sentences however do not have any apostrophe, this according to me is an attribute with most information gain. The below snippet of code helped me notice the differences between the number of apostrophe in Italian sentences and number of apostrophes in Dutch sentences.

```
• def hasApostrophy():
      dataset = open("train", encoding="utf8")
      english = []
      italian = []
      dutch = []
      en = it = nl = -1
      for line in dataset:
          words = line.strip().split()
          if words[0] == "en|":
              english.append(0)
              en += 1
          if words[0] == "it|":
              italian.append(0)
              it += 1
          if words[0] == "nl|":
              dutch.append(0)
              nl += 1
          for each in words[1:]:
              if (each.count("'") > 0 ) and words[0]
  == "en|":
                  english[en] += each.count("'")
              if (each.count("'") > 0 ) and words[0]
  == "it|":
                  italian[it] += each.count("'")
              if (each.count("'") > 0 ) and words[0]
  == "nl":
                  dutch[nl] += each.count("'")
      print(english)
      print("----------------")
      print(italian)
      print("----------------")
      print(dutch)
      threshold = 0
```

```
        val = 0
        for each in english:
            if each>threshold:
                val += 1
        print("english:", val,"lines")
        val = 0
        for each in italian:
            if each>threshold:
                val += 1
        print("italian:", val,"lines")
        val = 0
        for each in dutch:
            if each>threshold:
                val += 1
        print("dutch:", val,"lines")
        dataset.close()
```

- Has e: Yet again, most of the Italian sentences have a lot of e's, on making some changes to the above written code I was able to understand that e is the most common alphabet in Italian, the minimum number of e's the smallest sentence in Italian has are 4. On the contrary, most of the Dutch sentences have no e's at all. So, according to my algorithm if a line has more than 4 e's in a sentence, the feature's value will be set to True, otherwise, it will be set to False.

- Has a's: Like 'e', 'a' is another alphabet that is most common in Italian, this would be another major deciding feature. Similar to 'e', if the number of 'a's' in a line are greater than 4 then the feature's value would be set to True else, it would be set to False.

- Has KJWXY: From my research I found out that Italian words do not use K, J, W, X, and Y. So if the sentence contains none of the above-mentioned alphabets then it is MOSTLY (not definitely) a sentence written in Italian.

- Big word: This is a neutral feature, this feature was added to reduce over fitting. Most of the features I have chosen are very strongly biased towards one language. It is also possible for these features to be present in the other language as well but the algorithm might not be able to predict that because it has been trained too many times to predict one particular language. Using a neutral feature such as average word size helps avoid such a scenario. In my algorithm, if the average word size in a line is greater than 5 then it is considered to be a big word, and hence the value of this feature is set to True, else, it is set to False.

- Ends in O: Eyeballing through some Wikipedia random articles, I realized that very few Italian words end in 'O' or 'o', but a lot of Dutch words end with 'o' or 'O'. Writing a python code and finding the average number of words that end in 'o' or 'O' in Italian and Dutch made me realize that this could be another good differentiating feature.

All the features chosen above, have helped me achieve an 85.3% accuracy on decision tree. I haven't performed ada-boost algorithm, but I believe that using ada-boost will definitely help improve the accuracy of the algorithm.