MINI PROJECT -3 (INTRO TO AI)

EZHILMANI AAKAASH (40071067)

LANGUAGES USED:

- **English**: As per the project's requirement, the two e-books provided to train the english alphabets were put to use and the dump was successfully retrieved.
- **French**: Similarly, as per the requriement, the given two e-books were used for training and the dump was retrieved.
- **Dutch**: With the option of choosing any language of our choice (Same character set as English), I decided to experiment before finalizing on one.
    1. **Latin**: Viewed as the major contributor for the development of English, Latin seemed like a good choice for the 3rd option.
    2. **Italian**: With a character set of 21 alphabets, Italian is considered to be closely related to English.
    3. **Dutch**: Arguably the closest of all to English and French, Dutch has a character set of 26 alphabets.

RESULTS

With Italian, about 7 of the given inputs were classified wrong. (5 in Unigram and 2 in Bigram). This is primarily because of the missing alphabets in the italian character set. Despite the smoothing, the classification did not seem to improve to an acceptable level.

With Dutch and Latin being very similar in terms of structure, I decided to go with Dutch based on the langauge identification. With the training corpora I have for Dutch, classification seemed better in comparison to Latin.

This is primarily because of the quality of the corpora that is being used..
Dutch (4 Errors in Total including Bigram and Unigram/10)

Latin (7 Errors/10)

Therefore, I had decided on a set of 30 sentences based on each of the languages stated above and tested it.

1. With Italian – 7 Unigram Classifications were wrong and 6 Bigram
2. With Latin, 5 Unigram and 5 Bigram were wrong
3. With Dutch, 3 Unigram and 2 Bigram were wrong

This led me to conclude that the similarities between the these languages and French and English played a key role in deciding the language of the sentences. Latin seemed to cause the highest discrepancy.

SMOOTHING (WITH/WO)

- **WITHOUT:**

  Initially, the probabilities were calculated based on the dictionary filled with unsmoothed data/values.

  This meant that there was a high possibly that the input data/character combination may not be present.

  Since few combinations weren't present, the probability of the character(s) was declared to be zero. This caused a major impact while classifying the input sentences.

  Out of the 30 sentences given, 19 of them were either wrongly classified by both models or one of them. Around **36%** of accuracy.

  One of the biggest issues was the **classification of Dutch sentences**. With no smoothing, half the time, Bigram model seemed to mistake the sentence for English due to the fact that the grammar is similar and English word occurrences are more.

- **WITH**:

  Since finding the combinations that are missing AFTER the modeling is a huge task. Therefore, in-order to make things easier and quicker, I have initialized a dictionary (hash-map) with the combinations of the alphabets for a bigram.

  All the values have been set to zero. This way, the smoothing can be done over letters/combinations that aren't present in the dataset/corpora.

SMOOTHING

In order to smooth out the abnormalities, Laplace Smoothing equation was put into use and as per the requirements, the smoothing factor/Delta value was set to "**0.5**".

Once smoothing was done, better results were produced in terms of accuracy.

This was verified with in-depth analysis of each language and model.

For Unigram,

5 out of the 10 already given sentences were classified correctly.

For Bigram,

7 of the given 10 sentences were correctly predicted.

This is because, smoothing allowed the discrepancies (values with zero probabilities) to be assigned a minimum value causing the model to now make more accurate predictions. **This is visibly noticeable in case of French translations, where before smoothing was performed, the model classified most of the unavailable values as zero.**

Accuracy after smoothing, 63% which a good improvement considering the fact that the unsmoothed model was only providing a mere 36%.

- Upon increasing the delta values to up to 2, (0.9, 1.1, 1.8) there was much change in the output received.

OTprob_bi[k]=(OTmap1_bi.get(k)+0.5)/((singleCount.get(toCalculate[0]))+(len(OTmap1_bi.keys())*len(OTmap1_bi.keys())*0.5))

(For Bigram - With Smoothing)

ALLOWING FEW PUNCTUATIONS:

For experimentation purpose, I tried allowing the use of punctuation. This included the (‘) Accent, (_) Underscore, (-) Hyphen and (.) Fullstop.

This was done with the intention of allowing French Accent words and Dutch words to be classified better as these contained punctuation marks.

Unfortunately, this did not provide a better result than the already smoothed model and sometimes, even caused the accuracy to drop below 50% for various results.

Example:

Words such as In-order, Co-ordination etc caused the Unigram model to wrongly classify the sentences with these words as being French.

This is due to the presence of **Hyphen** in the French corpora.

In the given sentences, "**I'm OK.**" was classified as French by the Unigram model. This attributes to the fact that (') was allowed to be trained from the dataset/corpora.

THOUGHTS:

- With Ngram modeling, it seems like the training dataset/corpora plays a very major role. Although having more number of data sounds and really is beneficial, sometimes, as stated above, can cause some sort of hindrance for other language identification.
- Classification of the unseen data or instances is primarily based on the already-learned instances.
- Moving over to higher n values like trigrams does guarantee good results but only at the cost of extreme memory allocation. If the required resources are available, higher n values become possible.
- Lower n-value models are not suited for certain type of classification such as sentiment analysis. Having worked on it previously has given me few in-sights on this.
  Example:

  Using a Unigram model to perform sentiment analysis can quickly backfire. When it comes to words that express emotion or feelings, models with higher n-values are better suited.

  "Extremely Good"

  Unigram – Extremely (Neutral) , Good (Positive)

  Bigram – Extremely Good (Highly Positive)

REFERENCES:

- Theory Reference
  1. https://www.reddit.com/r/dataisbeautiful/comments/6fe871/frequency_of_letter_couples_bigrams_in_english_oc/?st=jp67cr6l&sh=549372c8
  2. http://isl.anthropomatik.kit.edu/cmu-kit/downloads/Letter_N_Gram_based_input_encoding_for_continoues_space_language_models.pdf
  3. http://okfnlabs.org/blog/2013/11/11/python-nlp.html

- Code References
  1. https://stackoverflow.com/questions/11011756/is-there-any-pythonic-way-to-combine-two-dicts-adding-values-for-keys-that-appe
  2. http://www.gutenberg.org/ebooks/18066?msg=welcome_stranger