Assignment Report : Language Identification

Arnob Mallik and **Arif Hasnat**
Department of Computing Science
University of Alberta

1 Methodology

At first, we converted the text files into character tokens by using the *list* method of python (tokens = list(text_contents))

We construct a class called *LanguageModel* which stores the *character tokens*, the *ngrams*, the *n_minus1_grams*, and the *vocabulary* of the language models. We used n values of 1 to 9. Note that, we need *n_minus1_grams* for calculating probabilities. For example, when we are working with trigrams, we also need bigrams for counting. The *vocabulary* is the count of unique tokens in a language which we needed for laplace smoothing

While training the language models, we also address the issue of unknown tokens. For a training file, we replaced the least common 2 tokens by our unknown token '#'. We had issues counting the frequencies of <UNK>. Hence, we decided to use '#' as our unknown token. We take the log2 of probabilities and calculate the perplexity as follows:

$$perplexity = 2^{-(log\_prob/character\_tokens\_count)}$$

The following subsections explain our model specific methodology for each of the 3 models:

1.1 Unsmoothed Model

For unsmoothed model we used the following equation to calculate log probabilities :

$$logprob = log2(\frac{tr\_tokens.count(te\_tokens[i])}{len(tr\_tokens)})$$

Here, *tr_tokens* is the list of unigrams from a particular training file and *te_tokens* is the list of unigrams from a test file. In the above equation we calculate the log probability of a particular unigram of a test file. We thus calculate the log probabilities for each of the unigrams of the test file and finally add them.

1.2 Laplace Model

For Laplace model, we use the following equation to calculate log probabilities :

$$logprob = log2(\frac{ngram.count(test\_token\_list) + 1}{n\_minus1\_gram.count(sliced\_list) + v})$$

Here, we used n values of 2 to 7. Suppose we are using n=3. In that case, *ngram* is the list of trigrams and *n_minus1_gram* is the list of bigrams from the training file.

test_token_list is a trigram from a test file and *sliced list* is the corresponding bigram of that particular trigram. For example, if ('b', 'i', 'g') is a trigram, then it's corresponding *sliced_list* will be ('b', 'i'). We need counts of both of the lists for our equation.

1.3 Deleted Interpolation Model

We construct a class called *InterpolationLanguageModel* which stores the *character tokens*, *gram_list*, *lambda_value_list* of the language models. For each training file, we store character tokens of the file, unigrams to nth-grams in gram list and *1st lambda* to *nth lambda* in *lambda_value_list*.

We generalized the deleted interpolation algorithm for higher values of n. At first, We calculate value of *lambda*s for each training file using that algorithm and store it to the language model for that training file as *lambda_value_list*.

Then, For each test file, we calculate probability with respect to each training file using *lambda_value_list* stored in language model for that training file by following equation:

P($t_i|t_{i-1}, t_{i-2}) = \lambda_3 P'(t_i|t_{i-1}, t_{i-2}) + \lambda_2 P'(t_{i-1}|t_{i-2}) + \lambda_1 P'(t_{i-2})$

Finally, we take the training file with minimum perplexity as our best guess.

2 Tuning the value of N

For unsmoothed model, we used n = 1. We did not use any other values of n to avoid 0 probabilities or infinite perplexities. For example, if we use bigrams in unsmoothed model, we might find a bigram in the test set such as ('a', #), where '#' is an unknown. In this case, if the bigram ('a', #) is not present in the training set, then we will have 0 probability.

For Laplace model, we ran our code for different values of n on the dev files and the results are shown in the following table:

n_value	correctly classified	avg_perplexity
2	53	8.92947707
3	54	8.682826242
4	54	10.627978018
5	54	13.2975456729
6	54	15.414428334
7	54	16.901163379

Note that, the *avg_perplexity* column shows the average perplexity of only the correctly classified dev files.

We had two criterias for selecting the best value of n :

- We pick the n values that classifies the highest number of dev files correctly
- Among those n values, we pick the one that has the lowest perplexity among the correctly classified dev files.

From the above table we can see that, among the n_values that classify 54 dev files correctly, n = 3 has the lowest perplexity. Hence, we picked **n = 3** as our n value for Laplace model. The test files are run using n = 3 for Laplace model.

Similary, for the interpolation model, we ran our code for different values of n on the dev files and the results are shown in the following table :

n_value	correctly classified	avg_perplexity
2	23	15.271912
3	42	10.015116
4	52	8.51751036
5	53	6.18733103
6	53	5.68579087
7	53	5.90025310
8	52	7.04831618
9	53	9.208777712

Similarly, we picked **n = 6** as our n value for Interpolation model.

3 Effect of Smoothing and N-Gram Settings

As we are using character n-grams and all the files contain English characters, unigrams do not carry much context. Hence, the unsmoothed unigram does not perform well in this case.

For Laplace and Interpolation model, we can see that the performance improves as we increase the value of n only until a certain point (n = 3 for Laplace and n = 6 for Interpolation). After that, the performance starts to decline. This is because of **overfitting**. As we increase the value of n beyond a certain point, the n-grams then contain too much context and overfit the training data.

Hence, it is important to find a balance between overfitting and underfitting.

Also, we observed that the perplexities of the Interpolation model are generally better than those of the Laplace model. This is because the interpolation model takes into account all the ngrams from 1 to n.

4 Error Analysis

While running our experiments, we observed that our models were struggling to classify two specific dev files. These two files were *udhr-deu_1901.txt.tra* and *udhr-deu_1996.txt.tra*. These two files were of the same language and had only slight differences, which explains the errors.

5 Contributions

All the tasks are performed after discussing with each other. We managed our code using GIT. So, both members have contributions in most of the tasks. But the main responsibilities were as follows:

Arnob Mallik	Arif Hasnat
Training Language Models	Command Line Arg Parsing
Unsmoothed	Perplexity Calculation
Laplace	Deleted Interpolation
Tuning n for Laplace	Tuning n for Interpolation
Report writing (partial)	Report writing (partial)

6 References

Chapter 3 & 8, Speech and Language Processing. Daniel Jurafsky & James H. Martin, 3rd Edition.