HW01 Project Report

1. Language Models

1.1 Question # a, b, c, d

The Brown corpus is a collection of 500 American English documents. They are spread across 15 categories like news, romance, fiction and other genres. The text in these documents are converted into lower case and then tokenized into 1161192(N) tokens using nltk word tokenizer. The vocabulary has 49815 samples. The percentage of hapax legomena occurrences are *1.8954660383468023* and the longest hapax is *nnuolapertar-it-vuh-karti-birifw-* where as percentage of dis legomena occurrences are *1.2383826275069068* and the longest one is *definition-specialization*. The graph for frequency(N_r) vs rank (r) plotted for the most frequent 50 words is shown in Figure 1. The respective log scaled version is plotted of samples whose $N_r > 0$ is shown in Figure 2.

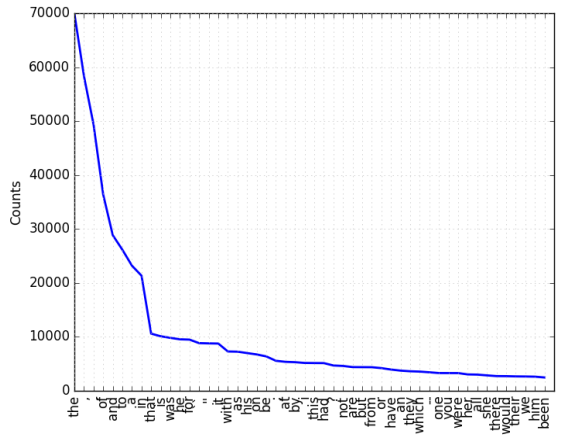

Figure 1 r vs Nr Plot

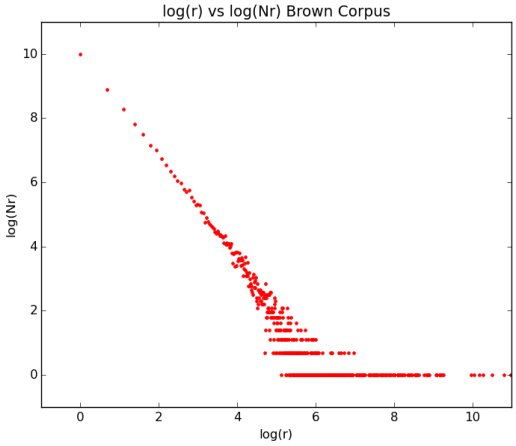

Figure 2 log(r) vs Log(Nr) Plot

Upon univariate linear regression on Figure 2 data using the objective function $\log(N_r) = a + b \log(r)$, The best possible values for "a" and "b" are obtained as [6.425370581363764; -0.914937359924159]. It is observed that the objective function that is used might not be a perfect fit for this data as it is not possible to estimate N_r for a new given sample (r) with good accuracy. This simple curve is best suited to estimate the frequency of most frequent words. The curve obtained using univariate linear regression is shown in Figure 3.

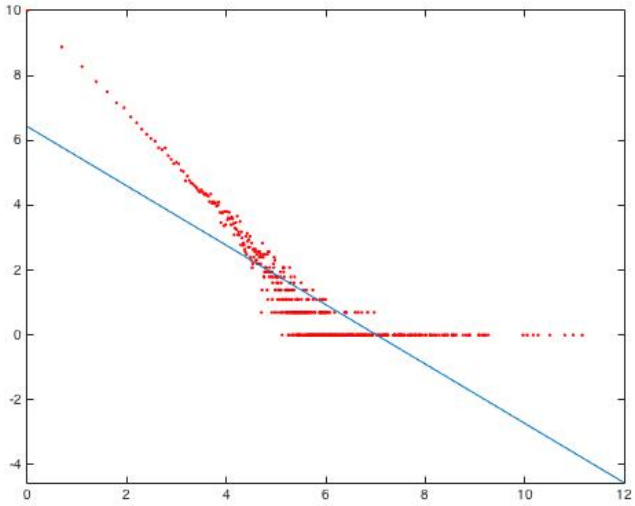

Figure 3 Univariate Linear Regression

1.2 Question # e

1.2.1 Data Preprocessing

The data is partitioned into train and test data sets. For testing, I have considered the first document (based upon the name) for each category whereas for training, I have used rest of the documents that are available on each category. I have created the vocabulary from the training data with words that appear at least twice by replacing all the hapax legomena's in the training data with <unk> token. To deal with Out of Vocabulary(oov) words, the oov words of the test data is replaced with <unk>. All the numbers in the train and test data are replaced with <num> token. Let's say the preprocessed train and test data as $XPRE_{Train}$ and $XPRE_{Test}$ respectively.

1.2.2 Methodology

I have trained a Unigram, Bigram and Trigram language models on $XPRE_{Train}$ using Katz back off, the discounted probabilities are computed using Lidstone smoothing with gamma = 0.2 Based upon the mix value, the words in $XPRE_{Test}$ are replaced by a random word of the vocabulary. Then compute the perplexity of $XPRE_{Test}$ on trained language models. So, for a given n gram model and mix value, the procedure is repeated 10 times to report avg. perplexity and std. deviation (STD). The experimental values are shown in Table 1.

		Mix %					
n_gram size		0	0.01	0.1	1	5	10
1	Mean	9.45953493	9.45953493	9.45953493	9.45953493	9.45953493	9.45953493
	STD	11.5855169	11.5855169	11.5855169	11.5855169	11.5855169	11.5855169
2	Mean	438.737767	434.718427	438.643603	436.748689	437.579681	438.918899
	STD	537.375332	532.439986	537.265515	534.921899	535.954206	537.570937
3	Mean	48.228723	48.3773357	48.9441875	48.8260272	48.4151098	48.3595107
	STD	59.0776154	59.2506266	59.9457821	59.8045804	59.2969207	59.2284749

Table 1 Avg. Perplexity and std.'s of different mixes

1.2.3 Results and Discussion

The 0% mix is just the $XPRE_{Test}$. From Table 1, for a Unigram model, the mean and STD remained same for all the mixes. For a Bigram model, the mean stays the same over mixes where as STD's does not. For Trigram model, there is there is very slight change of perplexity values for mean and STD among mixes.

2. Authorship Attribution

Given the writing's (X^{Author}_{Train}) of an authors and an unknown piece of text ($X^{?}_{Test}$), the task is to identify who possibly could have written it.

2.1 Data and Preprocessing

I have used a subset of the literature in nltk_data/corpora/Gutenberg for training and testing the language models to experiment on Authorship Attribution as shown in Table 2. For each authors' X^{Author}_{Train} I have created a vocabulary by replacing hapaxes with <unk> token as a pre processing step. Lets, say the preprocessed data as $XPRE^{Author}_{Train.}$

Table 2 Data Set Details

Author	**Training Data**	**Testing data**
Austen	Austen_persuasion.txt, austen_sense.txt	Austen_emma.txt
Chesterton	Chesterton_Thursday.txt, Chesterton_brown.txt	Chesterton_ball.txt
Shakespeare	Shakespeare_macbeth.txt, Shakespeare_caesar.txt	Shakespeare_hamlet.txt
Bible	-	Bible-kjv.txt

2.2 Methodology

There are multiple approaches to solve authorship attribution task. I have evaluated using the below two approaches using probabilistic language models. To address the sparsity issues, both the probabilistic distributions in both the approaches are smoothed using Lidstone smoothing.

2.2.1 Approach1

I have used an Intrinsic Evaluation method called perplexity. Perplexity is inversely proportional to Probability. So, upon using the authors' trained language models for computing the perplexity of the un known text $X^{?}_{Test}$, which ever model results in the least perplexity is the respective author who has written it.

2.2.2 Training

I have trained a trigram language model using Katz back off on $XPRE^{Author}_{Train}$ for each author. The gamma for Lidstone smoothing is 0.2.

2.2.3 Approach2

This approach computes the geometric mean of the probabilities of the singletons (say $XST^{?}_{test}$) present in $X^{?}_{test.}$ An equivalent way of doing this is compute the entropy of the $XST^{?}_{test}$ which uses log space for computations.

2.2.4 Training

I have trained a Unigram language model using X^{author}_{train} for each author. The gamma for Lidstone smoothing is 0.00001.

2.3 Results and Discussion

After training the language models and testing, I found both the approaches are useful to determine authorship attribution. The confusion matrices for both the approaches are shown in table 3 and 4.

Table 3 App. 1 Perplexity using Trigram LM

Test Data	Language Model			
		Austen	Chesterton	Shakespeare
	Austen	19.20	27.57	44.95
	Chesterton	27.65	24.49	50.24
	Shakespeare	46.79	54.10	25.93
	Bible	33.82	36.13	32.13

Table 4 App. 2 Entropy using Unigram LM

Test Data	Language Model			
		Austen	Chesterton	Shakespeare
	Austen	27.52	28.61	30.71
	Chesterton	29.47	26.72	30.67
	Shakespeare	30.26	29.35	27.40
	Bible	32.80	32.12	31.50

As shown in Table 3, When $X^{Austen}{}_{Test}$ is evaluated with 3 Trigram language models built on $X^{Author}{}_{train}$ for 3 authors, the least perplexity is observed for Austen which means Austen could be the author of it and he is. The same rule applies for rest of the authors. When $X^{Bible}{}_{test}$ is evaluated on 3 language models, the least perplexity is observed on Shakespeare's language model and this states that the writing style, language usage in the Bible are very similar to Shakespeare. The same trend is observed when a Unigram model is used for training and Singletons are used for testing. This could be because, Singletons capture the odd vocabulary choices that are very unique to an author or others. If the authors vocabulary choices are very similar, then the singleton approach might not yield very good results.

3. Language Detection

Given a piece of unknown text, the task is to find what language it belongs to?

3.1 Data and Preprocessing

I have used 11 language corpora available in ntlk_data/corpora/europarl_raw. Each language has a total of 10 documents out of them first 9 are used for training and the last one is used for testing. To make the language models case insensitive the data is pre processed to be in lower case. Along with it, I have also removed numbers and replaced multiple spaces with a single space. Let's call the preprocessed train and test data sets for a language as $XPRE^{Lang}{}_{Train}$ and $XPRE^{Lang}{}_{Test}$ respectively.

3.2 Methodology

In order to determine the language of given piece of text $XPRE^{Lang}{}_{Test}$, I have trained the language models as described below.

3.2.1 Training

Train a Bigram language model on sequence of characters using Lidstone probability distribution having gamma as 0.2

For each language

1. Break the train data into sentences using Punkt Sentence Tokenizer.
2. Break the words in the sentences of the training data to a sequence of characters.

3. Build a vocabulary of characters from the train data.

3.3 Evaluation

Given all the 11 trained language models and 11 $XPRE^{Lang}_{Test}$, I have evaluated the language models performance for language detection task. For each sentence in the $XPRE^{Lang}_{Test}$, compute its perplexity using the 11 trained language models. As we know that perplexity is inversely proportional to probability, classify the given sentence with least perplexity to the respective language. During Testing, when ever an out of vocabulary(oov) word is observed, the perplexity is reported as infinity. There are also chances that the bigram's count will be 0 which make the probability 0 in turn makes the computer report "Division by 0" exception while computing perplexity. So, to tackle this, I have smoothed language model using Lidstone Probability Distribution.

3.4 Results and Discussion

The confusion matrix (Table 4) shows the language detection performance on the given $XPRE^{Lang}_{Train}$ and $XPRE^{Lang}_{Test}$. The cell (L1, L2) contains the percentage of sentences from the test corpus of Language L1 that were classified to belong to L2.

Table 5 Confusion Matrix for Language Detection

Languages	English	Danish	Dutch	Finnish	French	German	Greek	Italian	Portuguese	Spanish	Swedish	# of Test Sentences
English	98.368	0.126	0.167	0.1255	0.544	0.1255	0	0.167	0.041841	0.2092	0.12552	2390
Danish	1.6088	91.45	2.992	0.1411	0.028	0.5645	0	0.056	0.084674	0.1693	2.90714	3543
Dutch	0.4593	0.026	97.86	0.051	0.153	1.0972	0.03	0.102	0.0255167	0.1786	0.02552	3919
Finnish	2.577	2.34	2.755	87.352	0.622	0.0889	0.03	3.14	0.5924171	0.1185	0.38507	3376
French	1.8929	0.118	0.177	0.562	95.03	0.0592	0.03	0.503	0.1183082	1.3901	0.11831	3381
German	0.3396	0.707	2.603	0.2547	0.057	94.199	0.03	1.443	0.1131862	0	0.25467	3534
Greek	0.0307	0	0.061	0	0	0.0307	99.8	0	0	0	0.03072	3255
Italian	0.4669	0.031	0.156	0.0934	0.591	0.0622	0.09	94.06	0.9337068	3.4858	0.03112	3213
Portuguese	2.6634	0.155	0.031	0.0929	0.588	0.0619	0	10.28	82.192629	3.9021	0.03097	3229
Spanish	0.756	0.091	0.091	0	1.089	0.0907	0	5.383	2.0260054	90.354	0.12096	3307
Swedish	1.354	6.383	1.713	0.1105	0.166	0.0829	0.3	0.967	0.1934236	0.2487	88.4775	3619

The language models are able to detect the sentences of language with good performance. Among all the languages, 99.8% of all Greek sentences have been classified correctly and this could be because Greek has got special unique characters like alpha, beta, rho etc. which are not available in other languages. The next best is English followed by Dutch and French. Comparatively, The Portuguese sentences are not very well detected as only 82.2% of them are classified correctly. The confusion matrix also captures the similarities between languages.