ANLP Coursework 1: Usernames: *S1424164*, *S1888641*

Q1.

```
def preprocess_line(line):
	line = re.sub('([^a-zA-Z0-9. ]|\n)', '', line)
	line = line.lower()
	line = re.sub('[0-9]','0',line)
	line = str('##'+line+'#')
	return line
```

Q2.

The hypothesis that we set forward is that a model with Laplace Smoothing applied was used with +1*X on the numerator and +30*X on the denominator, where X is some real number greater than zero. The probabilities were normalized by the sum of the conditional probabilities where the conditioning occurs on the previous two characters seen so far . The conditional probability for a particular character c_n, with previous characters c_{n-2}, c_{n-1} was calculated as $C(c_{n-2}, c_{n-1}, c_n)$ + 1*X divided by $C(c_{n-2},c_{n-1})$ + 30*X.

Using knowledge of English, we know that sequences starting with 'qx' are uncommon. These improbable[1] and out of vocabulary 3 and 2 letter sequences give us accurate fractional representations of the smoothing values as they likely have zero for both the counts of the numerator and denominator. Therefore we looked for the probability P(k|qx) and found a value of 3.333e-02. This value was matched by 48% of the entries in the list of probabilities, which concurs with the idea that many 3 and 2 letter combinations will be extremely rare in English corpora. Another piece of evidence to justify this hypothesis is that there exist common trigram letter sequences, with correspondingly uncommon 3 letter sequences. An example of this is P(' '|' 'b) which has a very common two letter sequence i.e. a space followed by b, yet the three letter sequence corresponding to observing the isolated character 'b' on its own, is uncommon. This will have the effect of increasing the denominator, yet keeping the numerator constant, thus reducing the size of the fraction relative to the zero case. This effect occurs yielding the probability of P(' '|' 'b) as 7.8e-05. Finally, by examining common 3 letter sequences that make up the majority of the corresponding preceding 2 letter sequences, we understood that some probabilities should be significantly closer to 1, though plus one smoothing decreases the weight of high probability events by a high degree. This can be seen for P(' '|do), which has a relatively high probability of 7.1e-01. Knowledge of English allows us to clearly understand that words beginning in d proceed with a vowel, mostly e, i or o. We believe that the # symbol indicates the end of a sequence if it occurs at the end of a three character string. Conversely, two characters denote ## both the beginning and end of a sequence respectively and is a special case used for modelling trigrams.

Q3.

To estimate the conditional probabilities, we calculated the counts in the training text, of all uniquely ordered three letter sequences divided by the counts of the previous two characters in the same sequence. The respective counts were stored in a dictionary data structure, mapping the letter sequences to corresponding counts. While the numerator initially had one added to it and the denominator thirty or twenty-nine[2], after re-computing perplexity, we

[1] By 'improbable' we mean strings that are not usually permitted by English spelling rules; while qu is a sequence permitted by English, q[^u] is usually not present in the English language. There may, however, be exceptions. Corpora can have spelling mistakes, abbreviations may contradict such intuitions, and rarely strings with q[^u] may be permitted (in strings such as Iraq, or other loanwords).

[2] We included twenty-nine results instead of thirty because we initialised our dictionaries with the set { #**, ##*, **#, *** }.

optimised the alpha value (justified in part six of this coursework). This approach falls in line with the definition of a trigram model with alpha-scaled Laplace Smoothing (Jurafsky and Martin, 2018). The alpha value was determined by plotting perplexity values for one thousand alpha values in the interval (0, 1) and choosing the optimal value. The justification for using smoothing was a) that we did not want to rule out the possibility of any obscure character sequences occurring and b) in order to account for the issue of sparse data. Probabilities of a sentence occurring, with such obscurities caused by potentially a typo or a new word from an open class category, will immediately assume a probability of zero. This means the model excludes the sentence from the language, which is an overly presumptuous claim. The equation that we used to calculate each probability is listed below:

$$P_{Laplace}(char_i|char_{i-2}char_{i-1}) = \frac{c(char_{i-2}char_{i-1}char_i)+\alpha 1}{c(char_{i-2}char_{i-1})+\alpha V}$$

A few simplifying assumptions were adopted. In particular, we only estimated the counts and corresponding probabilities for valid three letter sequences and we only parsed sentences greater than one character in length. In deciding the valid sequences, a regular expression was used such that trigram sequences would only be of the form (#**|##*|**#|***); the * character denotes characters corresponding to the allowed set '[a-z0.]', and '#' corresponds to the special literal which marks the beginning and ending of sentences depending on its position[3]. Invalid sequences have zero probability in our model and as such are not generated. The beginning of a line is denoted by '##'.

ng	0.796055979644	ngg	0.00089058524173	ngq	0.00089058524173
ng#	0.00216284987277	ngh	0.00089058524173	ngr	0.0123409669211
ng.	0.0263358778626	ngi	0.00216284987277	ngs	0.0212468193384
ng0	0.00089058524173	ngj	0.00089058524173	ngt	0.0136132315522
nga	0.00343511450382	ngk	0.00089058524173	ngu	0.00343511450382
ngb	0.00089058524173	ngl	0.00343511450382	ngv	0.00089058524173
ngc	0.00089058524173	ngm	0.00089058524173	ngw	0.00089058524173
ngd	0.00470737913486	ngn	0.00216284987277	ngx	0.00089058524173
nge	0.0861323155216	ngo	0.00725190839695	ngy	0.00089058524173
ngf	0.00216284987277	ngp	0.00089058524173	ngz	0.00089058524173

One would expect to find, that as English contains a large number of present participle words that end in 'ing ', the probability of 'ng ' is higher than a three letter sequence 'ngx'. This result is confirmed as 'ng ' has a probability three orders of magnitude higher that 'ngx'.

Q4.

a) Random sequence generated from the training.en file:

[3] While the sequences '##*' and '#**' only refer to the beginning of a line, '**#' only refers to the end.

```
##the the do porikesed of urs of than emplowe as sithe cated tzqapers actin the
pres up ing the db.egreporks is to of prommint hic .gxshs0gwjbtfforke fackway
commitimen##theres wevis whyqdrand tred ate dent then on twou to thatpjjusit
imps.##thes wor reposaff is an am of be efor euroactioncle ons dl
```

b) Generated from model-br.en probabilities:

```
##did thats tels thail.##hose.##whats pe.##sh slettors.##what wan a book
thave.##and aringue.##it cionny.##what.##sme.##wpwl.##willy do youtty.##wereme
drould thers.##ont thone.##whats this that at.##whattin.##tuq##what dide
toneephoseet thats tone.##ald bet to no are hats ther putes.##i a.##dragoin
```

How do the sequences differ?
The training.en file and model-br.en differ primarily in line length. The model-br.en lines are shorter than the training.en file and the words from model-br.en themselves are shorter than those generated from the training.en file. There are also more full stops ending the lines. The causes of this difference may be due to the difference of genre, format, model or subject matter of the training files that we used versus the training files used by model-br.en. It can also be noted that non-english strings do occur in both, primarily because of smoothing.

How does the code work?
The code generates the character sequence through a series of recursive calls such that at each call the probability of the next character is conditioned on the previous two characters. The probabilities are also normalised as the sum of the probabilities are sometimes not exactly one due to the imprecision of floating point arithmetic. The recursion is guaranteed to terminate as the variable `recursive step` which is a positive integer is decreased by one at each step. If a character ‘#’ is generated, which marks the end of a sequence, then an extra ‘#’ is generated to denote the start of the next sequence.

Q5.

Test file	Model training file	Perplexity value (bits per character)
data/test	English (data/training.en)	8.85769
data/test	German (data/training.de)	23.71622
data/test	Spanish (data/training.es)	23.29327

It is clear that perplexity is lowest for the model trained with English. When the training files have dissimilar trigram counts to the test file, the probabilities occuring will be proportionally lower, and thus perplexities will be higher. One cannot be sure that the test data is in English as one cannot easily establish a lower bound of the perplexity without rigorous testing. All the perplexity tells us is whether the training data fits with the test data and is more or less comfortable. One could imagine a disordered corpus with the same counts as English but no English words such as parts of the generated sentences. Potentially the perplexity could be a metric to distinguish between English and German or Spanish, but only if we were promised that the test data would be written in these languages. However, the perplexity values calculated have no precision metrics associated with them, and may differ across what we define as English (e.g. Twitter messaging vs. Shakespeare), thus it is unclear how much they vary within a language.

Q6.

Further question: What is the optimal alpha value (scaling factor of Laplace Smoothing) that will minimise perplexity given test data?

We addressed this question by iterating over 1000 α values[4] between 0 and 1, using intervals of 0.001, and then re-generating perplexity values, for the probability model with smoothing applied. We found that an alpha value around 0.7 gave the lowest perplexity results (see Figure 1). As the alpha value decreases, the probabilities become closer to that seen exactly on the training data. As alpha reduces below 0.6, the perplexity increases exponentially, as the probabilities are overfit to the training data. We also experimented with increasing alpha values. As alpha increases beyond 1, to 100, we saw the perplexity increase once more, representing underfit. Whether this result generalises to future documents, remains to be seen, subject to the style and subject of the text given.

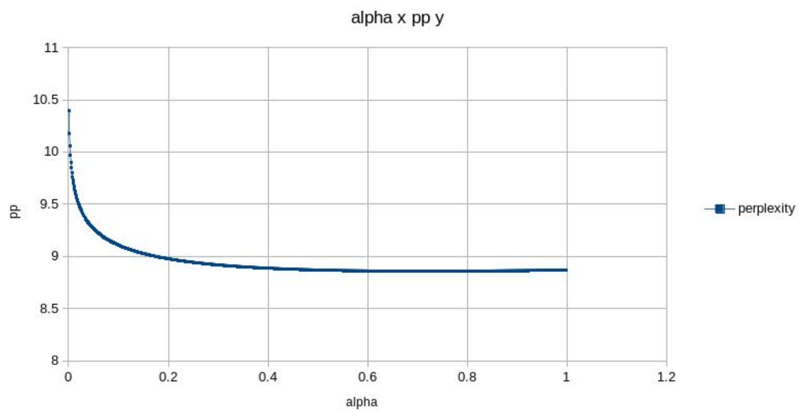

Figure 1. A graph showing the values interaction between perplexity and alpha after calculating perplexity values using the test set. In this graph, the alpha value that minimises perplexity occurs at $\alpha = 0.719$.

[4] Alpha is the scaling factor of Laplace Smoothing.

References

Goldwater, S. Accelerated Natural Language Processing Slides Week 2: ANLP Lecture 5-6., The University of Edinburgh http://www.inf.ed.ac.uk/teaching/courses/anlp/lectures/index.html#week2, 2018

Jurafsky D, Martin J., Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition., Third Edition draft., September 2018