ANLP Assignment 1

s1885540 & s1311612

1

```
Algorithm 1: preprocess_line(line)
  Require: line: a line of string from input file
  Ensure: a new line only with English alphabet, space, digits and dot character
  $rule \leftarrow re.compile('[\^\s.A-Za-z0-9]')$
  $line \leftarrow rule.sub('', line)$ ;                    // newline with only digit, alphabet, space and dot
  $line \leftarrow re.sub('\s\{2,\}', ' ', line)$ ;          // replace excess number of white spaces
  $line \leftarrow re.sub('\s[A-Z]\{2,\}\s', ' ', line)$ ;   // remove abbreviations
  $line \leftarrow re.sub('\s([A-Z]\{2,\}).', '.', line)$ ;  // remove abbreviations
  $line \leftarrow re.sub('[0-9]\{1,\}', '0', line)$ ;       // replace digits with a single zero
  $line \leftarrow line.lower()[:-1]$ ;                      // convert all characters to lower case
  $line \leftarrow '\#\#' + line + '\#'$ ;                   // add mark to specify start and stop
  return $line$
```

2

We can obtain information about the model's estimation method by looking at the probabilities assigned to unseen events. The assumption that the model had to deal with unseen events is based on the existence of phonotactical constraints. Native-level speakers implicitly know which phoneme combinations are possible in a given language. Although correspondences between graphemes and phonemes are not perfect, we can combine the knowledge that some character combinations are impossible with the observation that the model's trigram vocabulary includes all letter combinations (that some combinations involving non-letters are missing from the lexicon is not important in this case), and that none has been assigned a probability of zero. We can therefore conclude that Maximum Likelihood was not used as the estimation method. While some impossible trigrams may have been present in the corpus due to abbreviations or loanwords (which may disobey language-specific phonotactic constraints), it is extremely unlikely that it contained enough instances to fill an exhaustive list of trigrams with relative frequencies. Looking closer at impossible or intuitively highly unlikely trigrams we can draw further conclusions. Concerning trigrams starting with "xc" for instance, we can be reasonably sure that most of its continuations will have been unseen events (based on our intuition as speakers of English as well as the limited theoretical phonemic continuations of /ksk/ or /kss/). The model assumes a uniform distribution over all continuations ($\hat{P}(*|ng) = 0.033$), which would be an unlikely finding if the estimation method had taken lower level n-grams into account. Comparing for instance the intuitive bigram probabilities for "ca" and "cx", the former should occur much more frequently than the latter. Interpolation and back-off would have taken these bigram probabilities into account in the estimation of trigram probabilities so that we should have observed $\hat{P}(x|cx) < \hat{P}(a|cx)$. We therefore conclude that add-α smoothing was used as the estimation method as it does not take into account lower level n-grams and distributes probability mass from more to less frequent events resulting in all unseen events being assigned the same probability. Without seeing the corpus we cannot determine the value of α.

3

3.1 Description of our model's probability estimation method

We used interpolation to estimate trigram probabilities. Relative frequencies of trigrams ($P(w_3|w_1, w_2)$), bigrams ($P(w_3|w_2)$), and unigrams ($P(w_3)$) in our training set (80% of input data) were weighted and combined to estimate trigram probabilities. Weights ($\lambda_1, \lambda_2, \lambda_3$) were determined so as to minimize perplexity on a held-out set (10% of input data).

$$\hat{P}(w_3|w_1, w_2) = \lambda_1 \times P(w_3) + \lambda_2 \times P(w_3|w_2) + \lambda_3 \times P(w_3|w_1, w_2)$$

with

$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$

and where

$$\underset{\lambda_1,\lambda_2,\lambda_3}{\operatorname{argmin}} PP_{HELD-OUT}$$

3.2 Justification for choosing interpolation

Given that one of the aims of this model is the use of perplexity measures in predicting the language of a test document, we decided to use an estimation method that avoids overfitting the training data. As our interpolation hyperparameters were determined in order to minimize the perplexity of an unseen held-out set, we similarly expect to achieve low perplexity on other unseen sets of the same language. Interpolation further takes a character's context into account by combining bigram and unigram probabilities. This context is important because it allows the model to distinguish unseen combinations based on their components, thereby implicitly judging if a combination was unseen by chance or because it does not occur in a language.

3.3 Simplifying assumptions

As mentioned in question 2, languages are bound by phonotactics and although graphemes do not necessarily map onto phonemes, we acknowledge that there are impossible character combinations in a given language. Building a model that takes into account possible grapheme-phoneme correspondences and identifies impossible character combinations based on a full set of phonotactical rules would be beyond the scope of this project. Our model thus makes the simplifying assumption that all character combinations are possible. However, by taking a simple step during preprocessing we minimize the influence of impossible character combinations on our model. We remove all abbreviations from the input data as abbreviations constitute letter combinations that are not bound by the same rules as words that regularly occur in a language.

3.4 *ng* context trigrams: Explain what you would expect to find and why

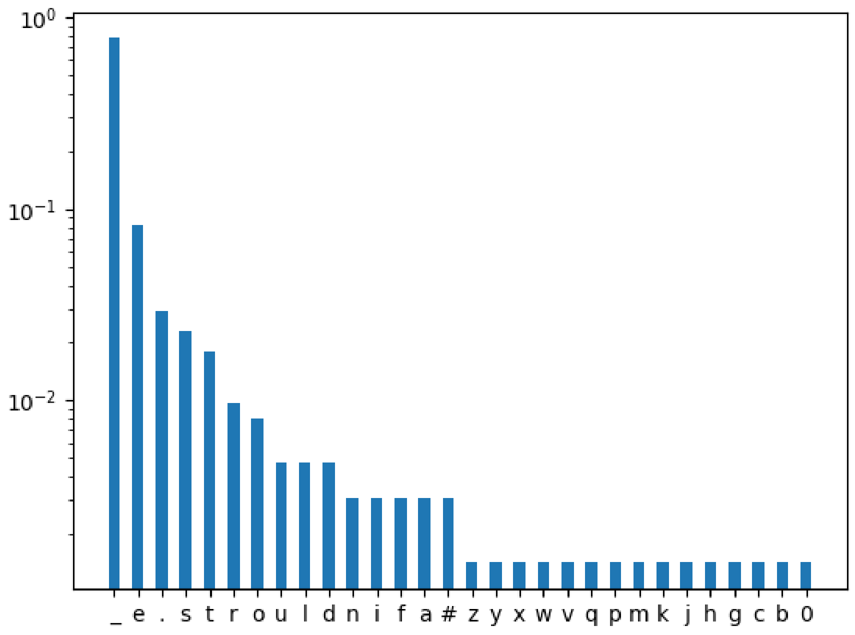
Given that "-ing" is a frequent inflectional and derivational suffix in English, and given also that "ng" is allowed in a coda position, we would expect "ng" to be followed by a word boundary more often than by a letter. Although the dot and hash characters can serve as word boundaries in our model, white spaces denote word boundaries most frequently. We would therefore expect that "ng" is most likely to be followed by a white space and that dot and hash characters show not the lowest probability. As we are looking at a full set of conditional probabilities (that is, all possible outcomes for the context "ng"), their cumulative probability should be 1. As our estimation method combines probabilities from trigrams with bi- and unigrams, we would further expect the probabilities for trigrams that include intuitively frequent bigrams (e.g. "ge", "gr") to be higher than those for trigrams including less common bigrams (e.g. "gx", "gk").

Table 1 provides an excerpt of our language model for English for all trigrams starting *ng*. The same information is displayed in a more intuitive format in Figure 1, with probabilities converted into log space and displayed in descending order.

Table 1: Estimated probabilities for trigrams with *ng* context

***ng* continuation**	$P_{LM}(< char > \vert ng)$	***ng* continuation**	$P_{LM}(< char > \vert ng)$
<space>	0.779	l	0.005
#	0.003	m	0.001
.	0.03	n	0.003
0	0.001	o	0.008
a	0.003	p	0.001
b	0.001	q	0.001
c	0.001	r	0.01
d	0.005	s	0.023
e	0.082	t	0.018
f	0.003	u	0.005
g	0.001	v	0.001
h	0.001	w	0.001
i	0.003	x	0.001
j	0.001	y	0.001
k	0.001	z	0.001

Figure 1: Estimated log_{10} probabilities for trigrams with *ng* context in descending order



3.5 Did these findings meet expectations?

As expected, *ng* was most likely to be followed by a white space. The general trend of vowels, fricatives and approximants being more likely continuations than stops matches our prediction that *ng** trigrams containing a relatively frequent *g** bigram will be assigned a higher probability than those with uncommon bigrams.

4

4.1 Explain briefly what the code does to generate a sequence given an LM

For the generator, we first divide trigrams into two classes: trigrams containing white space(s), dot(s) and # mark(s) are called **connectors** (e.g. 'is '), and trigrams containing only letters and 0 are called **inners** (e.g. 'his'). Intuitively, inner trigrams appear in a word, but connector trigrams appear between words. While transition happens between inners (shown in algorithm 2, line 11), we greedily select the next character with the highest probability according to the previous two, and if there are characters sharing the same highest probability, we randomly select one to break the tie. While transition happens between connectors, or connectors and inners (shown in algorithm 2,line 9), we randomly select the next character based on their probabilities, i.e. a candidate with higher probability is more likely to be selected. This random selection method is also called **Roulette Wheel Selection** [2].

```
Algorithm 2: $generate\_from\_LM(model, map, k)$
   Require: model: language model with trigram distirbution
   Require: map: key: all previous two bigram; value: set of next chars
   Require: k: required sequence length
   Ensure: seq: a generated sequence
 1 $seq \leftarrow \#\#$
 2 $cnt \leftarrow 0$
 3 while $cnt < k$ do
 4     $bigram \leftarrow seq[-2:]$
 5     $candidates \leftarrow map[bigram]$
 6     if $candidates \in \emptyset$ then
 7         $seq \leftarrow seq + \#\#$
 8         continue ;                                   // restart generator if no candidates
 9     else if $bigram.contains(\#\#, "", .)$ then
10         $next \leftarrow RWS(Probability_{candidates})$ ;          // RWS: Roulette Wheel Selection
11     else
12         $next \leftarrow Greedy(Probability_{candidates})$ ;  // Greedy:  select the highest prob candidate
13     end
14     $seq \leftarrow seq + next$
15     $cnt \leftarrow cnt + 1$
16 end
17 return $seq$
```

4.2 Random sequences of 300 characters generated based on our model trained on English corpus and model-br.en

Note that sequence start and stop markers "#" have been removed from the character count as they are auxiliary characters.

Our model	**Model-br.en**
##we ext not whis whis hat and the quident of lich and propeas und commis in gral the vot mand not be yould be the und be in and commis and propeas st whis vot dent hat the for mand whis gral propeas just and re for st dent the mand commis and and quident not the for be in the 0 0 in und not of the an	##right.###the do the the can judy.###now it it you do the bookay.###put it do you wan the the put kit do he the he it the for it the do do a kit like do you like a put one.###an you put gone.###he bookay.###an mome it it kit mome a the you right.###it it wan you do zippen he do the he bookay.###can you see for a put

4.3 Differences between the two sequences and possible causes

The main difference between the two sequences is the variety of trigrams found, with our model's sequence making use of a broader variety of character combinations than model-br's output. Model-br generates more character sequences that correspond to actual English words (e.g. "like", "gone", "you", "right", "okay"), while our model tends towards non-words that could be part of the English vocabulary but are not (e.g."commis", "vot", "propeas", "quident", "gral"). This could be due to overfitting of model-br to its training data, leaving some very frequent trigrams to assume high probability without taking into account continuations based on lower level n-grams. Indeed, when looking at the perplexity of model-br on the (English language) test file, its value is similar to that of our model when trained on Spanish or German and much higher than ours when trained on English, indicating that model-br overfits its training data in comparison to our model. This broader variety of combinations of our model compared to model-br further points to a possible difference in training data. Given the frequency of words (i.e. letter sequences enclosed by white spaces, dots or hash marks) in model-br's output that are commonly found in informal language (e.g. "you", "okay", "like"), we can make an educated guess that its training corpus is held in a comparatively informal register. Even without knowing that our training corpus consists of proceedings from the European Parliament, the almost complete lack of trigrams containing "you", the use of a broader range of combinations, as well as a longer word length indicates that it is held in a more formal style than model-br's training data. Finally, model-br produces more sequence start and stop markers than our model. A sequence restart occurs when no fitting candidates are found for continuation. The lack of sequence restarts in our model indicates again that it is less limited in its trigram continuation than model-br.

5

5.1 Perplexity of the test document under each of the three language models

Perplexity (PP) represents the inverse probability of a test sequence ($w_1 ... w_n$) given a language model (LM):

$$PP_{LM}(w_1 ... w_n) = 2^{-\frac{1}{n}\log_2 P_{LM}(w_1 ... w_n)}$$

$$PP_{EN} \approx 918$$

$$PP_{ES} \approx 2410$$

$$PP_{DE} \approx 2261$$

5.2 How can we use these numbers to guess the language of the test document without looking at it?

Given that our model uses equal trigram vocabulary sizes for all languages (all theoretically possible character combinations minus some pre-determined combinations containing special characters), we can directly compare the perplexity of models trained on different languages. As perplexity represents the inverse probability of the test data occurring given the model, a lower perplexity indicates a better fit between test and training data. In this case, the lower the perplexity, the more likely that test and training data come from the same language. The perplexity of the test data under each of the three language models strongly indicate that the document is written in English. Training our model on Spanish and German results in similar perplexity values ($PP_{ES} \approx 2410$, $PP_{DE} \approx 2261$), while the perplexity under the English model is considerably lower ($PP_{EN} \approx 918$).

5.3 If we had new test document and were told the perplexity under the English LM, would this be enough to know whether it was written in English? Why(not)?

While perplexity is useful in comparing the fit of different models to the same data, it does not provide meaningful measures when comparing different datasets. One point of concern is that differences in vocabulary sizes affect perplexity, with lower smaller vocabularies decreasing the model's perplexity. Although vocabulary size should not interfere with perplexity estimations under our language model because it operates with a closed vocabulary, we still have no way of accounting for intra-language variation between training and test data. As our training corpus was written in a formal register, we may assume that it will contain a rather varied repertoire of words (including e.g. technical and loanwords) that make use of a broader spectrum of character combinations than for instance vernacular English.

6 Does our generator improve when given information about phonotactic constraints?

As mentioned in question 3, our model makes the simplifying assumption that all character combinations are possible. However, languages show phonotactic constraints in that they favour certain sound combinations over others and disallow some combinations altogether. English syllables for instance are constructed based on a sonority hierarchy that postulates that sonority increases towards the syllable nucleus and decreases towards onset. Phonemes are classified through features such as voicing, which in turn determine their sonority [1]. Although graphemes do not necessarily map straightforwardly onto phonemes ("c" for instance can correspond to /k/ or /s/ and even more phonemes when combined with other graphemes), and despite the existence of more fine-grained phonotactical rules, we attempted to include at least rudimentary sonority constraints in our generator. We first classed graphemes into one of four classes based on the sonority features of what we judged their most common phonemic representation and in some cases on basis of where we hoped they would do the least damage (e.g. classing "j" and "w" as "−"), see Table 2. These classes aim to represent the fact that consonant clusters in onset and coda positions may be constituted of up to Non-letter characters were assigned to class <UNK>. The conditional bigram probability distributions over class X given class Y estimated from our training set function as weights in the generator. If the previously generated character for instance belongs to class "+", we weight the probabilities of all possible following letters (i.e. the trigram probability estimated by our character-based language model) by the conditional probability of their class occurring following a class "+" character. Table 3 contains output from our standard generator and our sonority weighted one (hash marks are removed for legibility).

Table 2: Generalized sonority classes for the purpose of generator improvement

Class	**Instances**	**Notes**
++	a e i o u	Vowels
+	l r	Approximants
-	f h v z s y	Fricatives
−	b c d g k m n p q t x w j	Stops and affricates
<UNK>	0 . <white space>	Non-letter characters

Table 3: Generator output: baseline compared to modified

Baseline Generator	**Sonority-modified Generator**
so yould ext in not the the propeas of ext commis commis not of the re and whis gral the the commis whis of gral of the and and .the the re in propeas knot in of gral for und the in the propeas xk of .the in und lich in quident xbould whis mand ext hat hat not of quident be mand st whis whis gral of	the and in juse the exament 0 and and hatin commin hatin foregion se exament se dent in witin mand hatin foregion que commin and mand noted the the on in on se poregion the region the 0 se commin and in witin the and hatin the on unitin in kociparegion younitin dent in unitin xibilicand .the se uniti

Comparing these sequences, we note at a first glance that our new generator produces longer words. Although most words produced are still non-words in the English language, they do not violate phonotactic constraints as opposed to some of the baseline generator's output (e.g. ”xbould”, ”xk”). Notably, the sonority-modified generator seems to prefer a simple consonant-vowel syllable structure. This does facilitate legibility but by avoiding consonant clusters it errs on the side of caution- the generator seems to favour precision over recall. One of the roots of this problem is in our classification system, which is an imperfect and overgeneralized imitation of phonemic representation and contains unequal number of character instances per class. In order to avoid this problem, character sequences could be converted into phoneme-level representations, these representations could then be classified based on sonority, passed into our generator and only in a last step converted back into graphemes.

References

[1] Philip Carr. *Modern Linguistics: Phonology*, chapter 9: Representations Reconsidered (i): Phonological Structure above the Level of the Segment. The Macmillan Press Ltd, 1993.

[2] Adam Lipowski and Dorota Lipowska. Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications*, 391(6):2193–2196, 2012.