INFO-Y-004: Natural Language Processing
Project report

Raphael Vander Marcken

March 22, 2018

1 Introduction

The goal of this project is to compute language models based on given training sets concerning some varieties of the English language and then using them in order to guess the language variety of a written document using a test set. The language model is built using trigrams letter. Given the generated models, generation of random outputs and language variety verification can be achieved. The program should be made using Python 3.

2 The problem

Before going further let us explain the problem in more details. Given a text, we would like to train a model in order to assign probabilities to the upcoming word/letter. This permits us to make predictions on the following word/letter given a certain text. The set of theses probabilities is what is called language model.
In order to achieve that we can compute the joint probability of words in sentence (or letters in words in our case) using the chain rule for probabilities. So given a sentence "This is a training sentence", we can compute the probability of the word "This" given that it is the first word, we can compute the probability of the word "is" given that we already have seen "This" and so on until the probability of the word "sentence" given that we have seen "This is a training". Of course, for this example all probabilities would be equal to 1 because we only have one sentence and we compute the probabilities by counting the occurences of the sentences with the current word divided by those without the current word but we should assume that we have a large number of sentences. Doing so is not efficient at all because we consider exact sentences and of course there are way too many possible sentences in order to compute the probabilities for every possible cases.
To resolve this issue, the Markov Assumption is used and is as follow : P(sentence | This is a training) $\tilde{=}$ P(sentence | training). This means that by taking into account the last word or last few words we have a pretty good approximation for this probability. It makes sense because quite often a word is commonly used after a small set (compared to the entire set) of words and we do not need the words appearing way before to predict the current one.
Using the Markov Assumption only with the last word to compute the model is called an Bigram model. By using the two last words, we call it a Trigram model, and so on. The same thinking can be applied for letters, were we only replace words by letter. In the case of this project we will consider trigrams of letter meaning that we will consider the two last letters in order to compute the probability of the upcoming letter.

3 Method

3.1 Core

The first thing to do to have a model on a given training set is to compute the trigrams. In order to do that, I compute the frequence of every possible bigrams we can make given the alphabet. In this case, the alphabet is the set of lowercase letters including the underscore symbol. After that, we need to compute the raw count of trigrams. To achieve that, I make a large matrix (using dictionaries to improve performance) where every row is one of the possible bigram and every column is one symbol of the alphabet. Every cell of the matrix will first contain the frequency of seeing the letter at the given column after the bigram at the given row or in other words the frequency of the trigram. After that we will compute the probability for every of those trigrams using the frequence of every possible bigrams computed earlier and using the Laplace smoothing where we take V = v^2 where v is the alphabet size in this case. After computing the probability for every trigram, we have our model ready and we can compute things like perplexity or generating random output according to the computed probabilities.

3.2 Simplifications

In order to facilitate the computations, some simplifications on the input text have been made.

- Every characters that were not ASCII alphabetic were removed.
- Every uppercase characters transformed into lowercase characters
- Every multiple successive whitespaces replaced by a single whitespace.
- Every single whitespace replaced by a double underscore.

The two last simplifications implies that in the end we have one long string consisting of only words separated by exactly two underscores.

Changing single whitespaces by double underscores serves as a separation between words. Because we are considering trigrams and if we use double underscores and we are considering the first/last letter of a single word then the probability for those particular trigrams will not be affected by the probability of the previous/next trigrams. As an example, let us consider the following sentence : "I love NLP", which after transformation gives "i__love__nlp". Then we will have to consider the following trigrams : [i__, __l, _lo, lov, ove, ve_, e__, __n, _nl, nlp]. We can observe that no final/first letter of any word are in the same trigram as a first/final letter of another word. So for a given trigram we know for sure that its probability is not affected by the end/start letter of another word.

4 Results

To test our model, a test text was supplied. Using it we can compute the perplexity score for each text and for each model. Using those scores we can compare them in order to get the predicted language variety. The perplexity score is an intrinsic evaluation, useful to test pilot experiments like ours. In our case, a good model is one that gives a high probability to the letter that actually occurs. Given that the perplexity is the inverse probability of the test set, normalized by the number of letters, and that we want the highest probabilities scores, then we want to minimum perplexity. When comparing the perplexity scores of the different models

on the same texts, the winner will thus be the one with the minimal score.
A list of such results is given on point 7, question 3 of this report with the winner noted in bold. We can observe that for 7 cases out of 9, we have a good prediction. Furthermore, for the two predictions that failed we can observe that the perplexity score for the models that should have win is always the second one so the error is not so large.

5 How to use the program

Execute the main.py file using one of the following commands at your disposal in order to compute the different tasks.

- By using the -c [AU|US|GB], you can compute the model of the given language variety. The model will be written in a file in the form model.X where X is the chosen variety.
- By using the command -g k -m [AU|US|GB] where k is an integer between 3 and 300, the program outputs a string composed of k letters formed using the probabilities of the given model.
- With the command -test [AU|US|GB], the user is prompted with the perplexity score of the chosen variety on every text from the test set.
- The -tests all command prompts the user with the perplexity score of every model applied to every test texts with yellow score indicating the minimum perplexity of all three models.
- The -t [AU|US|GB] command serves to generate all trigrams letters beginning with "iz" according to the supplied variety.

6 Discussions

- Do you need to run the test set on all three language models or is the score from a single variety model sufficient?

  The score alone does not tell much about the language variety of the text. Indeed a low perplexity means that we have high probabilities in the model and thus that predictions should be good. But given that we run our models on the same language (English) and the differences will not be high enough to tell the language variety with the score alone.

- Would a unigram or bigram language model work as well? Explain why (not).

  Let us first consider the unigram case. In this case we can summarize the probabilities in the model being influenced only by the frequency of a given letter. For example, if we would model a language containing a lot of occurences of the letter "e" using unigrams, the probability for this letter will be high. In practice, to differentiate between language variety using unigrams would be very inefficient. But if we were to distinguish between languages that uses very different letter frequencies, unigrams would be sufficient.

  Concerning language modeling using bigrams, we would get reasonable results but not as good as with trigrams. It would work quite well with language varieties that have a lot of

words that differentiate only by one letter. For example, it is known that in US English, there exists more words using a "z" instead of a "s" for the same words comparing to the GB English and thus using bigrams to differentiate between those two should get reasonable results.

- Do the language models show anything about similarity of the varieties? Why (not)?

  If we were to compare the probabilities of the same trigrams between two models, we could obtain a difference. After that we could compute the mean difference between all the trigrams composing the models. That result should be a good indicator of the similarity of the varieties. Because if two probabilities of the same trigram does not differentiate much, this means that when encountering the related bigram, the next letter should be the same. So the lesser the difference, the more similar the varieties are.

- Can you think of a better way to make a language variety guesser?

  A first way I can think of is to consider trigrams of words instead of letters. This way we could spot language expressions that are unique to some variety. Although we would miss varieties differences that are in the letter level (like a tendency to use more "z"'s instead of more "s"'s for same words). We could then use some weighting techniques to use both trigrams of words and letters in order to mix them to refine the probabilities.

  Another way is to start from the idea that such a guesser would work with very similar languages except for very small differences. We could then imagine a guesser that, for exactly similar words or expressions we do not pay very much attention (ie. small weight) but in cases where we have very similar words or expressions containing only a very small difference, we pay a lot of attention to it (ie. big weight). In the end, when analyzing a text, those weights will be attributed to each different language varieties by summing them and the prediction would be the one with the maximum weight.

7 Tasks

- An excerpt of the language model for British English and another excerpt for American English model, displaying all n-grams and their probability with the two-letter history i z (E.g. izo, ize, izo etc.).

  US :
  iza : 0.1616924820551568
  izb : 0.0007555723460521345
  izc : 0.00037778617302606723
  izd : 0.00037778617302606723
  ize : 0.443520967132603
  izf : 0.00037778617302606723
  izg : 0.00037778617302606723
  izh : 0.0007555723460521345
  izi : 0.055156781261805815
  izj : 0.00037778617302606723
  izk : 0.00037778617302606723
  izl : 0.00037778617302606723
  izm : 0.001511144692104269

izn : 0.0007555723460521345
izo : 0.024178315073668303
izp : 0.00037778617302606723
izq : 0.00037778617302606723
izr : 0.00037778617302606723
izs : 0.00037778617302606723
izt : 0.0007555723460521345
izu : 0.009822440498677748
izv : 0.00037778617302606723
izw : 0.00037778617302606723
izx : 0.00037778617302606723
izy : 0.00037778617302606723
izz : 0.017378163959199094
iz_ : 0.012844729882886286

GB :
iza : 0.10870907967881409
izb : 0.0006176652254478073
izc : 0.0006176652254478073
izd : 0.0018529956763434219
ize : 0.33662754786905497
izf : 0.0006176652254478073
izg : 0.0012353304508956147
izh : 0.0018529956763434219
izi : 0.028412600370599134
izj : 0.0006176652254478073
izk : 0.0006176652254478073
izl : 0.0006176652254478073
izm : 0.0006176652254478073
izn : 0.0006176652254478073
izo : 0.02347127856701668
izp : 0.0006176652254478073
izq : 0.0006176652254478073
izr : 0.0006176652254478073
izs : 0.0012353304508956147
izt : 0.0006176652254478073
izu : 0.00926497838171711
izv : 0.0006176652254478073
izw : 0.0006176652254478073
izx : 0.0006176652254478073
izy : 0.0012353304508956147
izz : 0.0253242742433601
iz_ : 0.017912291537986413

Observation :
If we were to sum those probabilities we could observe that the result would be higher in the US case than in the GB case. This is explained because it is known that the US language tends to replace the "s" with a "z" more often than in the GB language. This is why we have a higher probability to see trigrams like "ize" in the US language ($P \simeq 0.443$) than in the GB language ($P \simeq 0.336$).

- 200 characters of random output for each of the three languages varieties.

  AU : l__unin__es__of__ovesidembillic__suggleamet__me__cabotaincessfruniste
  __or__covin__of__giventurp__lishis__he__mand__of__of__sounicautheorl
  itan__parded__bar__st__abol__wo__thavediffech__gram__hat__be__b

  GB : jkware__keente__gaties__a__in__sts__what__smoutenagrear__tons__i__wh
  otreve__the__and__ty__deast__licen__hanand__men__spopeund__zincifire__ban
  zyjvyzphouden__wit__pay__sccum__ch__bold__offents__and__but

  US : lf__arever__ther__of__them__cals__the__ed__dowle__aduk__to__fur_
  _but__theirsuess__the__of__ust__revid__piesshe__or__scus__is__of__hav
  en__hing__of__his__to__s__of__juseenow__food__ective__ser__sitted__

  Note :
  I kept the double underscores simplification to easily distinguish the whitespaces but one should keep in mind that to get the "correct" results, those double underscores should be replaced by a single whitespace.

  Observation :
  We can first observe that for most of the words have a reasonable length that tends to prove that the probabilities for the whitespaces have a good approximation. For the words itself, it is rather normal that for the majority they do not mean anything given that we only consider trigrams of letters and not words. It is better to interpret those results by looking at them in group of three letters. Nevertheless we can observe that we obtain some "real" words like [them, of, a, in, to, fur, the, and, food, ...]. In fact, the smaller the word the bigger the probability we have to have an existing word because as explained just before, if we are in the case of a long word, we should just consider group of 3 letters and thus the bigger the words, the more group of 3 letters we have and the more chance we have that the final word would not exist.

- The perplexity scores from the three language models for each test sentence.

  TEXT nr 0 | EXPECTED RESULT = AU | Perplexity for : **AU : 6.424558154862745**
  TEXT nr 0 | EXPECTED RESULT = AU | Perplexity for : GB : 6.706362400204314
  TEXT nr 0 | EXPECTED RESULT = AU | Perplexity for : US : 6.640202183651292

  TEXT nr 1 | EXPECTED RESULT = AU | Perplexity for : **AU : 6.406748416058941**
  TEXT nr 1 | EXPECTED RESULT = AU | Perplexity for : GB : 6.675019182274301
  TEXT nr 1 | EXPECTED RESULT = AU | Perplexity for : US : 6.6409537665194

  TEXT nr 2 | EXPECTED RESULT = AU | Perplexity for : **AU : 7.143498482313628**
  TEXT nr 2 | EXPECTED RESULT = AU | Perplexity for : GB : 7.4311516099969435
  TEXT nr 2 | EXPECTED RESULT = AU | Perplexity for : US : 7.198212372188335

  TEXT nr 3 | EXPECTED RESULT = GB | Perplexity for : AU : 5.729379335812877

TEXT nr 3 | EXPECTED RESULT = GB | Perplexity for : **GB : 5.602016273917891**
TEXT nr 3 | EXPECTED RESULT = GB | Perplexity for : US : 6.1659738367452634

TEXT nr 4 | EXPECTED RESULT = GB | Perplexity for : AU : 6.39017953299862
TEXT nr 4 | EXPECTED RESULT = GB | Perplexity for : GB : 6.212592459105628
TEXT nr 4 | EXPECTED RESULT = GB | Perplexity for : **US : 6.205505809395692***

TEXT nr 5 | EXPECTED RESULT = GB | Perplexity for : AU : 5.907477804860285
TEXT nr 5 | EXPECTED RESULT = GB | Perplexity for : **GB : 5.900361336970806**
TEXT nr 5 | EXPECTED RESULT = GB | Perplexity for : US : 5.938902866085299

TEXT nr 6 | EXPECTED RESULT = US | Perplexity for : AU : 5.564627621271685
TEXT nr 6 | EXPECTED RESULT = US | Perplexity for : GB : 5.289588538426385
TEXT nr 6 | EXPECTED RESULT = US | Perplexity for : **US : 5.246940511275261**

TEXT nr 7 | EXPECTED RESULT = US | Perplexity for : AU : 6.288582484733484
TEXT nr 7 | EXPECTED RESULT = US | Perplexity for : GB : 6.289976093628364
TEXT nr 7 | EXPECTED RESULT = US | Perplexity for : **US : 6.244166993527736**

TEXT nr 8 | EXPECTED RESULT = US | Perplexity for : AU : 5.47890800206603
TEXT nr 8 | EXPECTED RESULT = US | Perplexity for : **GB : 5.224499493063011***
TEXT nr 8 | EXPECTED RESULT = US | Perplexity for : US : 5.23771967915472

Note :
A star is added at the end of the perplexity score for those with a wrong prediction.

- Precision across all test sentences.
  The precision is 7/9 correct answers.