Arbitrary Precision and Arbitrary n-grams

Alec Story and Thomas Levine
avs38 and tkl22

February 27, 2011

Sentence Generation

For sentence generation, we did not include our unknown processing, and do not convert the first occurrence of each word to an unknown token.

Each subsection contains two examples from the Shakespeare corpus and two from War and Peace. These corpora were chosen because they were determined at rough glance to be the easiest to parse and had the fewest problematic.

Unsmoothed

Unigrams

> Bring in Take made the pass a Caesar in gold well leave
>
> Exeunt . hope , and Milan your A Will thy have Come the . . advantage , together your either to '
>
> He fell other along " therefore , and had about He chief shorten now him life . something
>
> I had transference shrugged man with you The lit . to lift the her had he . and have the hand individuals dragging the , place been to Natasha in inside . revelry hide nation room beaming eager all of the the , remounts to

Bigrams

> Enter LYSANDER , and you lie dull actor on my speech , an ordinary pitch , Loyal and full of our tongues , see , That which physic to my horse for her youth and sealed bags Of base earth ; Though twenty several mistress will you have done To have found .
>
> Silence , who .
>
> The horses ' familiar to the army , envy and looked questioningly : "You won't be thus depriving him of the estate with his will send me forty thousand men in choosing her governess kept saying good-by .

I have begun to Tushin and yet further battles , the man in the defense of that temple , they were standing beside whom you seen in the battle of dry branches of meeting it has brought the smile Berg with a mocking , and even impossible to be long paced angrily .

Trigrams

How - traitor ?

The Greeks upon advice , Hath alter'd that good wisdom Whereof I know by their christen names , Unless thou tell her so .

No , but it can't be helped It happens to the first to arrive , Princess , I think .

Rostopchin , coming to Petersburg , look out at night his feet .

Smoothed

Unigram

th ! come Lear things of the I

me . I

they its intimate just such But There the them said shout irresistible always heard none with and of dinner had To

day good-natured Secondly same , joyful on talking the appeared advantage . unreasoning men by

Bigram

Hector .

I arrest thee here , which time , that his fault , and blows .

But my wife should have nobody and ask the enemy was too .

Say it's farther .

Trigram

She would have made fair work !

Come , my lord , I'll bring you to please his Majesty , Herod of Jewry dare not say he is much abus'd with tears thou keep'st command .

All that he had no news from the chase .

If I decline the honor of the bushes .

Trigrams lead to more comprehensible sentences, which is to be expected, and bigrams are barely useful. The unigrams also seemed to produce longer sentences; it's possible that trigrams and bigrams lead to a progression from words that are likely at the beginning of sentences to words that are likely at the end, and then to end the sentence, despite sentence ends being fairly infrequent. Additionally, *War and Peace* seems to produce longer sentences (particularly when observed over more iterations than can be reproduced comfortably here), probably because the sentences in the original are longer, so sentence ends are rarer.

Our n-gram generator only treated sentence endings by adding a single symbol, '#', and did not break the input into individual sentences, nor do we break the perplexity input into sentences in the same manner.

Arbitrary Precision

One of the two extensions we chose was to implement the probability and perplexity calculations using arbitrary precision arithmetic (we also implemented it using logarithmic arithmetic). This can lead to slower processing, but in our profiling, most of the time was spent in dictionary manipulation rather than mathematics. This approach guarantees that there is no data loss until we move into processing floats during exponentiation.

We used python's +gmpy+ package to represent the probabilities as arbitrary rational numbers (python's integers are already arbitrary-precision), and +mpmath+ to perform the final exponentiation to calculate the perplexity (the n-th root). The libraries could only compute non-integer powers of integers or floats, and integer powers were rounded to the nearest integer, and therefore, inaccurate, so (very high precision) floats provided the best results. This final stage of computation of perplexity was the only point where imprecise mathematics was used.

Our hypothesis was that the arbitrary precision math would reveal a small amount of error in computing the perplexity logarithmically.

Arbitrary n-grams

Because the computations were general enough, we were able to easily abstract the code to produce count, smoothed count, and probability tables for arbitrary n-grams. We also compute all of the smaller n-grams, which are required for generating or analyzing the beginning of a sequence of words, and for other computations.

Large n-grams (particularly $n > 4$) tend to result in whole Shakespeare sentence fragments. For example, "O , my offence is rank , it smells to heaven ; It hath the primal eldest curse upon't , A brother's murther !" was generated by a hexagram, and is a direct quote from Hamlet, act 3 scene 3. This is a well-documented phenomenon, and is due to the fact that it's quite possible that the preceding $n - 1$-gram appears only once in the entire corpus, so the sentence generator has no choice but to choose the next word in the quotation.