CSE156 - PA1 Report - Xinmeng Li

Language Model Implementation

- Roadmap

I designed a trigram model combining the smoothing strategy of laplace and linear interpolation. On the path towards this final version of trigram model, I went through the process that unigram without unk -> unigram with unk -> trigram with unk but no smoothing -> trigram with unk and laplace smoothing -> trigram with unk and linear interpolation -> trigram with unk and the combination of these two smoothing techniques.

- Basic Algorithms

The calculation of mle follows the formulas on the lecture slides: q(wi) = count(wi)/count(all of the words in the training corpus), q(wi|wi-1) = count(wi-1,wi)/count(wi-1), q(wi|wi-2,wi-1) = count(wi-2,wi-1,wi)/count(wi-2,wi-1).

The way for unk is that replacing all of vocabulary that only appear once in the training corpus with unk. For example, for a sequence (u,v,w), if v only appears once, we will save (u,unk,w) in the dictionary instead of (u,v,w), and at the same time increment the count of (u,v,w) to that of (u,unk,w) to keep a valid distribution. Note that this belongs to the preprocessing that occurs before the computation of mle.

Each model has respective dictionary for the maximum likelihood estimate (be logged with base 2) of unigram, bigram and trigram. As a result, it is convenient to get access to all of the parameter values when it is necessary, such as linear interpolation, whose computation need qmle for q(wi), q(wi|wi-1), and q(wi|wi-2,wi-1).

- Model Framework

The model also has a poplist, which is the list of words that only appeared once in the training corpus. When we calculate perplexity, to get the conditional probability of a sequence (u,v,w), we need to first check if any word in the sequence is in the poplist. If the answer is yes (let's say v in poplist), we will compute the conditional probability of (u,unk,w) instead. I also check if there is any word not existing in the training corpus. Let's say w is not in the training vocabulary, thus we use (u,unk,unk) instead. Finally, let's check if (u,unk,unk) is in the keys of trigram. If it is, we are able to read the log conditional probability value directly from the trigram dictionary; If it's not, in the laplace smoothing, we assign this whole sequence to a unigram called UNK to represent all sequences not existing in the training corpus. We initialize the value of this UNK to be the number of occurrences of all of the words in the poplist before the mle computation. I have checked that this UNK only has such a slight effect on the distribution that the whole distribution is still valid. (i.e. (sum over the counts of all of the trigram sequences+count for UNK)/(sum over the counts of all of bigram sequences+number of words in the training corpus) = 1). This is not a very good approach as it obviously ignores the context of the trigram sequence. Compared with laplace, linear interpolation extracts and utilizes more information from an unknown sequence. In the model with linear interpolation, if we want to find the log conditional probability given a trigram sequence (u,v,w), we also first do unk preprocessing, and then check if the preprocessed sequence is in the trigram keys. If it is, we return (lambda3*q(wi)+lambda2*q(wi|wi-1)+lambda1*q(wi|wi-2,wi-1)); If it is not, we have to set lambda1 to be 0 in the case that (wi-1,wi) is in the bigram keys or set both lambda1 and lambda2 to be 0 if (wi-1,wi) is not in bigram keys.

- Turing Hyperparameters

In my trigram model with laplace smoothing, I found out that with the alpha decreases, the perplexities for train and validation dataset decrease. For example, with alpha = 0.01, perplexity of train is around 240 and that of dev is around 65; with alpha = 0.001, perplexity of train is around 10 and that of dev is around 44. I chose 0.01 as the optimal. Here I also observe that when alpha >= 0.01, the perplexity of train is greater than that of dev, which is abnormal. I think this is because the conditional probability of UNK is too large, so I tried several values to scale it. For the scale constant = 0.1, 0.01, 0.001, the perplexities for train and dev are around (290, 410), (350, 2557), (423, 15957) respectively. Clearly, 0.1 is the optimal.

The lambda values also need to be tuned in the linear interpolation. In the case that the sequence is in the trigram training corpus, I have tried (lambda1,lambda2,lambda3) = (0.8,0.1,0.1), (0.7,0.2,0.1) and (0.6,0.3,0.1). In the case that (u,v,w) not in trigram corpus but (v,w) in bigram corpus, I have tried (lambda2,lambda3) = (0.8,0.2), (0.7,0.3) and (0.6,0.4), and finally choose (0.8,0.1,0.1) and (0.8,0.2).

- Issue

The model with linear interpolation has a relatively small train perplexity, for example, I got around 16 for train but 381 for dev. I thought about two possible issues: first is that the distribution is probably be off. But my unigram, bigram and trigram are valid respectively. Based on the deduction talked about in the lecture, as long as lambda1+lambda2+lambda3 =1 and lambda1,lambda2,lambda3>=0, the new estimate should define a distribution. Another possible issue is that there might be mistakes in my conditional probability calculation. I have reflected on my algorithms and cannot figure out what’s going wrong. I guess that’s because the data is sparse, so I combined laplace with linear interpolation and the result now seems to make sense. Please feel free to comment if you have any suggestions, thanks!

Analysis of In-Domain Text

- Unigram with unk

```
----------------------
brown  read. train: 39802 dev: 8437 test: 8533
total UNK is  14349
sum of p(w): 1.0000000000001779
vocab: 17227
train: 886.7326210620298
dev  : 661.3054058443886
test : 658.6142906338957
----------------------
reuters  read. train: 38169 dev: 8082 test: 8214
total UNK is  10648
sum of p(w): 0.9999999999998426
vocab: 15410
train: 1016.4380307825963
dev  : 859.134696374125
test : 856.5198628865861
----------------------
gutenberg  read. train: 68740 dev: 14729 test: 14826
total UNK is  7002
sum of p(w): 1.0000000000000435
vocab: 12439
train: 509.5952721508509
```

dev : 507.96308405015736
test : 515.0466190496704

- Trigram with Laplace Smoothing

----
brown
total UNK is 17763
vocab: 578551
train: 372.94952566479816
dev : 569.4867873408967
test : 570.1312713742814
----
reuters
total UNK is 14248
vocab: 628353
train: 208.82598428586292
dev : 426.0229991514343
test : 429.4876318687648
----
gutenberg
total UNK is 18319
vocab: 1021281
train: 301.2156483569697
dev : 648.5064054175745
test : 651.2823800123917

- Trigram with Linear Interpolation

----
brown
train: 19.604328435730643
dev : 546.6876534763542
test : 549.6107100393208
----
reuters
train: 17.049300063185566
dev : 152.01705123355507
test : 155.2384366847628
----
gutenberg
train: 24.58611078985109
dev : 280.75314567287677
test : 283.11422810226964

- Trigram with Linear Interpolation & Laplace Smoothing

----
brown
train: 427.686

dev :568.916
test : 574.937
----
reuters
train: 309.465
dev : 386.078
test : 386.448
----
gutenberg
train: 277.156
dev : 345.166
test : 349.965

We observe that all of the models outperform the unigram model. The linear interpolation model has a gap between the perplexity of train, and test and dev dataset. The laplace model performs well with perplexities around hundreds. The linear interpolation & smoothing model has average train,dev and test perplexity than other models.

The top 20 trigram sequence with the highest mle in brown, reuters and gutenberg are listed respectively.
[('the', 'end', 'of'), ('the', 'White', 'House'), ('part', 'of', 'the'), ('*', 'One', 'of'), ('UNK', 'and', 'UNK'), ('Mr', 'and', 'Mrs'), ('the', 'number', 'of'), ('*', 'In', 'the'), ('*', 'This', 'is'), ('One', 'of', 'the'), ('UNK', 'on', 'the'), ('members', 'of', 'the'), ('the', 'fact', 'that'), ('UNK', 'in', 'the'), ('UNK', 'of', 'the'), ('some', 'of', 'the'), ('*', 'It', 'is'), ('as', 'well', 'as'), ('one', 'of', 'the'), ('the', 'United', 'States')]
[('Inc', 'said', 'it'), ('*', 'He', 'said'), ('CORP', 'lt', 'UNK'), ('the', 'end', 'of'), ('The', 'company', 'said'), ('03', '09', '87'), ('87', '03', '09'), ('INC', 'lt', 'UNK'), ('is', 'expected', 'to'), ('UNK', 'lt', 'UNK'), ('04', '09', '87'), ('mln', 'Nine', 'mths'), ('3RD', 'QTR', 'NET'), ('UNK', '1ST', 'QTR'), ('the', 'United', 'States'), ('Nine', 'mths', 'Shr'), ('UNK', '3RD', 'QTR'), ('mln', 'Avg', 'shrs'), ('he', 'said', 'END_OF_SENTENCE'), ('QTR', 'NET', 'Shr')]
[('as', 'soon', 'as'), ('it', 'shall', 'be'), ('which', 'the', 'LORD'), ('the', 'men', 'of'), ('*', 'Oh', 'END_OF_SENTENCE'), ('tabernacle', 'of', 'the'), ('LORD', 'God', 'of'), ('LORD', 'your', 'God'), ('And', 'it', 'came'), ('the', 'sight', 'of'), ('the', 'tribe', 'of'), ('And', 'thou', 'shalt'), ('the', 'name', 'of'), ('the', 'house', 'of'), ('the', 'hand', 'of'), ('LORD', 'thy', 'God'), ('it', 'came', 'to'), ('the', 'sons', 'of'), ('the', 'children', 'of'), ('the', 'son', 'of')]
The top 20 bigram sequence with the highest mle in brown, reuters and gutenberg are listed respectively.
[('through', 'the'), ('does', 'not'), ('may', 'be'), ('rather', 'than'), ('will', 'be'), ('kind', 'of'), ('per', 'cent'), ('from', 'the'), ('should', 'be'), ('during', 'the'), ('in', 'the'), ('part', 'of'), ('able', 'to'), ('of', 'the'), ('at', 'the'), ('It', 'is'), ('New', 'York'), ('on', 'the'), ('number', 'of'), ('United', 'States')]
[('added', 'END_OF_SENTENCE'), ('4TH', 'QTR'), ('He', 'said'), ('sources', 'said'), ('lt', 'UNK'), ('QTR', 'NET'), ('mths', 'Shr'), ('compared', 'with'), ('expected', 'to'), ('due', 'to'), ('did', 'not'), ('United', 'States'), ('CORP', 'lt'), ('Nine', 'mths'), ('09', '87'), ('NET', 'Shr'), ('INC', 'lt'), ('1ST', 'QTR'), ('3RD', 'QTR'), ('Avg', 'shrs')]
[('dare', 'say'), ('ark', 'of'), ('sort', 'of'), ('inhabitants', 'of'), ('obliged', 'to'), ('pray', 'thee'), ('round', 'about'), ('Frank', 'Churchill'), ('midst', 'of'), ('into', 'the'), ('Captain', 'Wentworth'), ('spake', 'unto'), ('out', 'of'), ('Oh', 'END_OF_SENTENCE'), ('sons', 'of'), ('tribe', 'of'), ('son', 'of'), ('able', 'to'), ('according', 'to'), ('children', 'of')]
These words indicate the style of each corpus clearly: brown for common English, reuters for financial news and gutenberg for literature.

Since it takes like forever to sample a sentence from the whole trigram training corpus, I hand construct a few sentences and measure its log probability.

I first write a sentence using words from top 100 highest mle, then replace some words with words from top 9900 to 10000.
Brown
['*','If','you','is','one','of','member','of','the','United','States'] the log prob is -65.335927830711
['*','Dr','Bonner','is','one','of','member','of','University','of','Oklahoma','in','the','past'], the log prob is -94.83004760995267

For the other two domains it would be similar.

Analysis of Out-of-Domain Text

- Unigram

----
x train

	brown	reuters	gutenberg
brown	736.265	598.823	543.101
reuters	572.381	916.936	495.611
gutenberg	465.843	466.315	523.656

----
x dev

	brown	reuters	gutenberg
brown	661.305	596.061	539.588
reuters	568.179	859.135	499.483
gutenberg	465.25	467.371	507.963

----
x test

	brown	reuters	gutenberg
brown	658.614	595.189	545.026
reuters	572.652	856.52	495.366
gutenberg	469.235	466.88	515.047

- Trigram with Laplace Smoothing

x train

	brown	reuters	gutenberg
brown	372.95	444.897	560.483
reuters	811.982	208.826	801.28
gutenberg	880.889	750.255	301.216

----
x dev

	brown	reuters	gutenberg
brown	569.487	447.035	558.551
reuters	811.045	426.023	801.52
gutenberg	879.285	757.544	648.506

----

x test

	brown	reuters	gutenberg
brown	570.131	444.758	560.007
reuters	814.587	429.488	796.455
gutenberg	876.699	744.663	651.282

- Trigram with Linear Interpolation

x train

	brown	reuters	gutenberg
brown	19.6043	568.079	547.176
reuters	671.982	17.0493	638.299
gutenberg	627.821	597.778	24.5861

----

x dev

	brown	reuters	gutenberg
brown	546.688	570.529	544.765
reuters	668.958	152.017	647.431
gutenberg	623.395	607.429	280.753

----

x test

	brown	reuters	gutenberg
brown	549.611	567.367	545.978
reuters	680.427	155.238	637.567
gutenberg	623.293	589.279	283.114

- Trigram with Laplace Smoothing & Linear Interpolation

x train

	brown	reuters	gutenberg
brown	427.686	452.263	508.824
reuters	485	309.465	446.308
gutenberg	369.123	293.462	277.156

----

x dev

	brown	reuters	gutenberg
brown	568.916	455.492	508.844
reuters	479.936	386.078	451.665
gutenberg	367.18	296.509	345.166

----

x test

	brown	reuters	gutenberg

brown	574.937	450.702	506.765
reuters	483.813	386.448	446.19
gutenberg	370.762	293.348	349.965

We observe that model trained on brown generalized well on both reuters and gutenberg, probably because the language of no matter in which field such as finance or literature, besides a small portion of terms, is close to present day normal American English. Reuters neither performs well on brown nor on gutenberg, which indicates that the professional language of financial news is usually restricted to economic field and thus hard to generalize on other kinds of corpus. Gutenberg does not generalize well on the rest of domains, as it contains literature, which does not include words used in daily conversation or financial newspaper.