Language Modeling

Statistical Natural Language Processing - PA1

Saideep Reddy Pakkeer

University of California San Diego

15.04.2019

INTRODUCTION

Language modeling is one of the basic and important components of our study - Natural Language Processing. In this exercise of language modeling we'll use n-gram approach to model and also explore different aspects like smoothing, filtering etc. Specifically, we will work with trigram language model with add k smoothing and linear interpolation. Additionally we'll compare with the baseline unigram language model for evaluating the performance (using perplexity)

DATA

We use three different corpora for our implementation as well as for the analysis we carry out in this exercise. Here is a brief description of what they represent

CORPUS	DESCRIPTION
Brown Corpus [1]	Standard corpus to represent the present-day American English
Gutenberg Corpus [2]	Selection of text from public domain works by authors including Jane Austen and William Shakespeare
Reuters Corpus [3]	A collection of financial news articles that appeared on the Reuters newswire in 1987

LANGUAGE MODEL IMPLEMENTATION

Our task for this exercise is to implement a trigram language model and use appropriate smoothing or interpolation to generalize the model better. Later we compare how it performs across different corpora

Trigram language model

It is a probabilistic language model where we assume the probability of occurrence of each word depends only on the preceding two words (2^{nd} order Markov approximation) and we take the maximum likelihood estimate to model the conditional distribution. So, we model each sentence as

- $p(x_1,...,x_n) = \prod_{i=1}^{n} q(x_i \mid x_{i-2}, x_{i-1})$ where $q(w_i \mid w_{i-2}, w_{i-1}) = \text{count}(w_{i-2}, w_{i-1}, w_i) / \text{count}(w_{i-2}, w_{i-1})$ &

And our vocabulary is $|V|$ = {all the words in the corpus}$\cup${'END_OF_SENTENCE'}

If we come across any unseen word we can give a naive-backoff probability (0.000001 in our case) or $1/|V|$. As you can see this is clearly a very bad idea to give ~0 or no probability to unseen words. We take care of this by smoothing or interpolation which we'll see next. Just to compare we've also implemented the unigram (which assumes independence from any preceding word).

Additive/Laplace smoothing

In additive/laplace smoothing we try to redistribute some of the probability from the seen trigrams to unseen ones. The way we do it is by adding $|\delta|1$ to the denominator and $\delta|V|$ in the denominator. So, our new estimate for conditional probability becomes

$q(w_i \mid w_{i-2}, w_{i-1}) = \frac{count(w_{i-2}, w_{i-1}, w_i) + |\delta|}{count(w_{i-2}, w_{i-1}) + \delta|V|}$. It is called add one smoothing when $\delta = 1$

What this basically does is, it distributes the probability of unseen trigrams a probability of $\frac{1}{|V|}$ rather than a zero. Sometimes, it assigns too much weight, hence we scale it by δ

Linear Interpolation

In linear interpolation we introduce three parameters $\lambda_1, \lambda_2, \lambda_3$ and make our maximum likelihood estimate $q(w_i|w_{i-2}, w_{i-1}) = \lambda_1 q_{ML}(w_i|w_{i-2}, w_{i-1}) + \lambda_2 q_{ML}(w_i|w_{i-1}) + \lambda_3 q_{ML}(w_i)$ where $\lambda_1 + \lambda_2 + \lambda_3 = 1$. The condition why all lambdas have to sum up to 1 comes from the fact that our new estimate should define a distribution. We are going to experiment with different lambdas and see how our model performs on all the different corpora.

There are other methods like discounting for example Katz Back-Off Model where you take some probability off the seen words and redistribute the missing probability in a particular fashion. In this exercise I haven't experimented with Katz Back-Off Models, so we'll analyze laplace smoothing and linear interpolation.

Hyperparameter tuning

For our analysis we evaluate the language model using perplexity which is out measure of language model "goodness". It is defined as

Perplexity: $2^{-(\frac{1}{M}\sum_{i=1}^{m} log_2 p(s_i))}$,where M=|V| & $p(s_i)$ is the prob of each sentence

Laplace Smoothing:

In case of laplace smoothing we experiment with different values of δ and see it's effects on the perplexity. We make use of the dev set in each of the three corpus to compare and evaluate the model performance. Although we look at the training perplexity, we are interested in the validation/dev perplexity to determine the best δ for laplace smoothing.

Linear Interpolation:

In this case our hyperparameters would the values of lambda itself. So, we do a grid search and evaluate the performance on different values of lambda. We will see how the perplexities vary in the coming sections.

Analysis of In-Domain Text

In this section we see the perplexities when trained and validated on the same corpus. We do that for each of the corpus to evaluate the performance. Apart from smoothing I also tried to replace the low frequency words with "UNK".

Perplexity

	Perplexity without smoothing (no 'UNK')			Perplexity without smoothing & words with freq<3 replaced with 'UNK'		
Corpus	Train	Dev	Test	Train	Dev	Test
Brown	5.75	793.45	824.00	6.84	362.24	369.99
Reuters	6.07	128.24	134.09	6.59	84.22	86.09
Gutenberg	9.14	209.66	213.48	10.00	141.75	143.73

After experimenting with a higher frequency(5) to replace with 'UNK' we get even lower perplexity but clearly the train perplexity increases as seen above. But it is clear that it is performing way better on training set than the dev and test set although the perplexity has certainly improved after replacing the highly infrequent words with 'UNK'. But make no mistake - these perplexities are not an indicator of how well our model is performing because, this is only tested on the in-domain corpus. This model performs very bad for out-of-domain corpus (perplexity in the range 5000-10000).

We will see linear interpolation along with its performance in out-of-domain texts in the out-of-domain analysis.

Comparison with Unigram model

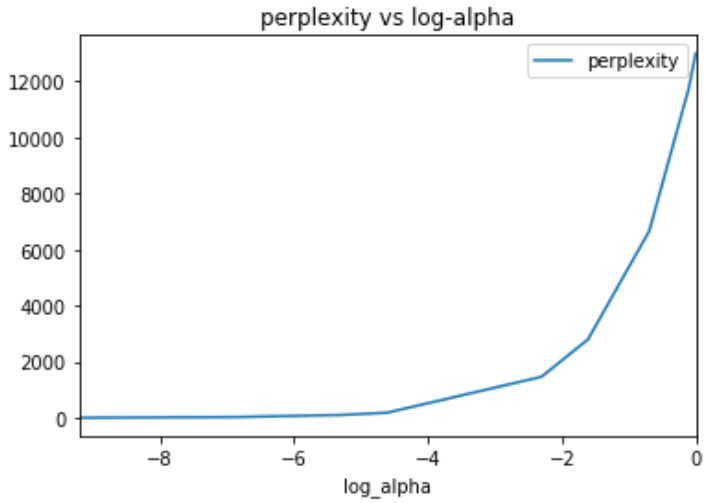
Next we try laplace smoothing and also compare with unigram model that was already built. I replace the infrequent words with 'UNK' in all further analysis

	Unigram			Trigram with laplace smoothing ($\delta = 0.01$)		
Corpus	Train	Dev	Test	Train	Dev	Test
Brown	1513.80	1589.38	1604.19	189.09	353.55	378.17
Reuters	1471.20	1479.09	1500.69	117.14	192.02	204.65
Gutenberg	982.57	991.50	1005.78	158.85	263.39	279.97

The perplexity values get even better if we tune the δ to even lower values but clearly it's going to overfit as you've seen earlier. Let us see now analyze the perplexity varies as we tune it with the dev corpus. We will also see that though the whole sentence doesn't make sense, the phrases in them mean better compared to the unigram model.

Performance with different hyperparameters

The perplexity is similar to the below graph for all the corpora when tested on the in-domain text. Below is for the Brown corpus.



	dataset	perplexity	alpha
0	brown_train	7.998141	0.0001
9	brown_train	30.939805	0.0010
18	brown_train	106.111425	0.0050
27	brown_train	189.091522	0.0100
36	brown_train	1467.987823	0.1000
45	brown_train	2797.972136	0.2000
54	brown_train	6664.735219	0.5000
63	brown_train	11730.362350	0.9000
72	brown_train	12990.151450	1.0000

	dataset	perplexity	alpha
1	brown_dev	14.143492	0.0001
10	brown_dev	58.378312	0.0010
19	brown_dev	199.200852	0.0050
28	brown_dev	353.554340	0.0100
37	brown_dev	2722.976882	0.1000
46	brown_dev	5187.259577	0.2000
55	brown_dev	12356.420830	0.5000
64	brown_dev	21753.690360	0.9000
73	brown_dev	24091.260300	1.0000

	dataset	perplexity	alpha
2	brown_test	14.792639	0.0001
11	brown_test	61.926183	0.0010
20	brown_test	212.661568	0.0050
29	brown_test	378.176355	0.0100
38	brown_test	2924.731385	0.1000
47	brown_test	5576.345177	0.2000
56	brown_test	13297.458820	0.5000
65	brown_test	23426.198120	0.9000
74	brown_test	25946.514980	1.0000

You observe a similar pattern when tested on the other two corpora. (Details on github)

Sample sentences

Here are few examples of sampled sentences (temperature = 0.6). If we tune the temperature even lower we get 'UNK' sampled as we have replaces all the infrequent words together as UNK. I just extracted few phrases from the sampled sentences that seem good. For complete sentences (which are long to include in the report) take a look the github repo in references where I included them.

Brown Corpus

- "It is burdensome midway preponderating motels …", "ceaselessly expects prune …"

Reuters Corpus

- "The company said it has acquired Chatsworth AUTO..", "it will be coloured migrating Horner reference thoroughly.."

- "And the sons of Gad Coward landward Faculties goodman …", "But you contenders sutes…"

Analysis of Out-of-Domain Text

Perplexity & Comparison with Unigram model

	Without smoothing			Linear Interpolation (λ_1 =0.6, λ_2 = 0.2, λ_3 = 0.2)			With laplace smoothing		
Corpus	Brown_train	Reuters_train	Gutenberg_train	Brown_train	Reuters_train	Gutenberg_train	Brown_train	Reuters_train	Gutenberg_train
Brown	5.75	12445.2	2403.68	7.97	1764.4	443.54	215.17	567.22	554.81
Reuters	7016.02	6.07	18264.2	587.99	14.20	499.54	457.21	175.67	472.60
Gutenberg	3100.91	44706.9	9.14	104.51	307.65	15.68	332.60	353.23	183.31

As we can see the out-of-domain performance is very bad without smoothing. With smoothing it is better and with additive smoothing it becomes much better. With proper hyperparameter tuning both the smoothing techniques prove to perform much better than without smoothing on out-of-domain text.

Corpus	Brown_train	Reuters_train	Gutenberg_train
Brown	1513.80	6780.82	1758.06
Reuters	3806.39	1471.21	4882.80
Gutenberg	2616.57	12420.1	982.57

Unigram out-of-domain perplexity

Performance on different corpora and results

Trigram with laplace smoothing:

	X train			X dev			X devtest		
Corpus	Brown	Reuters	Gutenberg	Brown	Reuters	Gutenberg	Brown	Reuters	Gutenberg
Brown	215.17	567.22	554.81	362.69	501.67	512.62	362.54	523.81	530.21
Reuters	457.21	175.67	472.60	392.41	231.35	472.05	418.39	230.12	468.72
Gutenberg	332.60	353.23	183.31	324.70	354.36	231.48	328.13	350.87	230.39

Here each entry represents the perplexity when trained on the corpus on the left and tested on the corpus on the header. For example, 472.05 is the perplexity when trained on Reuters and tested on dev set of Gutenberg. One thing is clear from the table i.e. all the model perform well on the same domain compared to out-of-domain text which is expected. One interesting thing to notice is the last column, where

the Gutenberg perplexity is more than double when it is trained on different corpus. One reason could be that both Brown and Reuters are modern compared to Gutenberg (which includes texts from Shakespeare)

ADAPTATION

One scheme that we can use to combine a subset of B's corpus to train with A's corpus to adapt to the corpus of B is to take the most frequent trigrams/bigrams/unigrams in B's corpus that are not in A's corpus and include them along with the corpus of A. So, we can experiment with different fractions of B's vocabulary and see the performance of the model on B's validation/test corpus.

The perplexity of Reuters corpus when the language model trained on Brown corpus is used: **293.70**

And the perplexity of Reuters corpus when it's own training data was used is: **69.92**

Corpus	Perplexity
5% of Reuters corpus in Brown	193.89
10% of Reuters corpus in Brown	176.07
20% of Reuters corpus in Brown	155.19
40% of Reuters corpus in Brown	138.16

Clearly, as we keep adding the vocabulary from B to A, the perplexity keeps decreasing.

CONCLUSION

We have explored the n-gram language model and different smoothing techniques like laplace smoothing and linear interpolation. Smoothing techniques ensured that the probability mass is distributed to unseen n-grams and performs better when tested on out-of-domain text. We have also seen how we can adapt our language model by using a fraction vocabulary of another domain's corpus. Clearly looking at the sampled sentences and their meaning we see that n-gram models (even with smoothing) are a very bad choice to model language. We are not looking at the context beyond a few words preceding the word, the parts of speech of the word while modeling the language although we know that they are important aspects of the language modeling.

REFERENCES

1. https://github.com/saideepreddy91/Statistical-NLP/tree/master/A1_cse256_sp19
2. http://www. hit.uib.no/icame/brown/bcm.html
3. http://gutenberg.net/
4. https://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection