SRILM - (toy-dataset)

Training data.

```
Training data:

<s> I am Sam </s>
<s> Sam I am </s>
<s> Sam I like </s>
<s> Sam I do like </s>
<s> do I like Sam </s>
```

Test data - 1

(a) <s> Sam I do I like </s>
(b) <s> Sam I am </s>

test data - 2

<s> Sam I do like linguistics </s>

Count of Unigram

<s> = 5
I = 5
am = 2
Sam = 5
do = 2
like = 3
</s> = 5

Total = 27

Probability of Unigram

P(<s>) = 5/27
P(I) = 5/27
P(am) = 2/27
P(Sam) = 5/27
P(do) = 2/27
P(like) = 3/27
P(</s>) = 5/27

Probability of bigram.

P(I | <s>) = 1/5	P(am | I) = 2/5	P(Sam | am) = 1/2
P(Sam | <s>) = 3/5	P(I | Sam) = 3/5	P(like | I) = 2/5
P(<do> | <s>) = 1/5	P(I | do) = 1/2	P(do | I) = 1/5
P(like | do) = 1/2	P(</s> | Sam) = 2/5	
	P(</s> | am) = 1/2	
	P(</s> | like) = 2/3	

#1 Using a bigram language model based on the above training data, compute probability and perplexity of the following sentences? Repeat this with Unigram language model and observe the differences. Which one would you rate the best language model?

(a) <s> Sam I do I like </s>
(b) <s> Sam I am </s>

Bigram

(a) $P(\text{sam}|\langle s\rangle) \cdot P(I|\text{sam}) \cdot P(\text{do}|I) \cdot P(I|\text{do})$

$\cdot P(\text{like}|I) \cdot P(\langle /s\rangle|\text{like})$

$= \frac{3}{8} \cdot \frac{3}{5} \cdot \frac{1}{8} \cdot \frac{1}{2} \cdot \frac{2}{5} \cdot \frac{2}{3} = \frac{9}{125} \cdot \frac{2}{15} = 0.009648$

$\log_{10} P(s) = -2.0155627$

$PP(s) = \sqrt[6]{\frac{1}{P(s)}} = 2.16734$

$PP_1(s) = \sqrt[5]{\frac{1}{P(s)}} = 2.52995$

(b) $P(\text{sam}|\langle s\rangle) \cdot P(I|\text{sam}) \cdot P(\text{am}|I)\ P(\langle /s\rangle|\text{am})$

$= \frac{3}{5} \cdot \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{1}{2} = \frac{9}{125}$

Hence according to language modelling

(b) is more probable (better)

$\log_{10} P(s) = -1.4266$

$PP(s) = \sqrt[4]{\frac{1}{P(s)}} = 1.093048$

$PP_1(s) = \sqrt[3]{\frac{1}{P(s)}} = 2.40374$

Unigram (Note i have take <s> & </s> both in vocab)

(a) $P(<s>) \cdot P(\text{sam}) \cdot P(I) \cdot P(\text{do}) \; P(I) \cdot P(\text{like}) \cdot P(</s>)$

$= \frac{5 \times 5 \times 5 \times 2 \times 5 \times 3 \times 5}{(27)^7}$

(b) $P(<s>) \; P(\text{sam}) \; P(I) \cdot P(\text{am}) \; P(</s>)$

$\frac{5 \times 5 \times 5 \times 2 \times 5}{(27)^5}$

So, by comparing unigram & bigram model, we found bigram is much better model than unigram.

Because
- Probability of sentence is very low
- Perplexity will be very high.

Hence bigram is good :)

#2 Using a bigram language model based on the above training data, compute probability and perplexity of the following sentence? Repeat this with Unigram language model and observe the differences.

```
<s> Sam I do like linguistics </s>
```

Here.

$P(\text{linguistics} \mid \text{like}) = 0/3 = 0$

$P(\langle s \rangle \mid \text{linguistic}) = \frac{0}{0}$

$P(S) = P(\text{sam} \mid \langle s \rangle) \times P(I \mid \text{sam}) \times P(\text{do} \mid I) \times P(\text{like} \mid \text{do})$

$\times P(\text{linguistic} \mid \text{like}) \times P(\langle /s \rangle \mid \text{linguist}) = 0$

Perplexity $= \infty$

Now - Smooth bigram

$$P_{\text{laplace}}(w_i \mid w_{i-1}) = \frac{[C(w_{i-1} w_i) + 1]}{C(w_{i-1}) + V}$$

Note. $|V| = 6$ (size of vocabulary)

(a) $P(\text{do} \mid \langle s \rangle) = \frac{2}{11}$

(b) $P(\text{do} \mid \text{sam}) = \frac{1}{11}$

(c) $P(I \mid \text{Sam}) = 4/11$

(d) $P(I \mid \text{do}) = 2/8$

(e) $P(\text{sam} \mid \langle s \rangle) = 4/11$

(f) $P(\text{like} \mid I) = 3/11$

(g) $P(\text{sam} \mid \text{do}) = 1/8$

(i) $P(\text{linguistics} \mid \text{like}) = \frac{1}{11}$

(j) $P(\langle /s \rangle \mid \text{linguistic}) = 1/8$

$$P(S) = P(\text{sam}|\langle s\rangle) \times P(I|\text{sam}) \times P(\text{do}|I) \times P(\text{like}|\text{do}) \times P(\text{linguistic}|\text{like}) \times P(\langle/s\rangle|\text{linguist})$$

$$= \frac{4}{11} \times \frac{4}{11} \times \frac{2}{8} \times \frac{3}{11} \times \frac{1}{11} \times \frac{1}{8}$$

$$\text{Perplexity} = \sqrt[5]{\frac{1}{P(S)}} = PP(S)$$

Here, we compare smooth & unsmooth, then we find, unsmooth has infinite perplexity but on other hand smooth have finite perplexity.

Language modelling using SRILM

Aman Kumar 17025

17th October 2020

Brown Corpus

1 Why did you get -inf probability for OOV words above? What was the significance of -unk and -addsmooth 0 ? How can you fix it?

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

- We get inf probability for OOV words because count of that word is 0 , which make probability 0 or undefined.

- -unk take tag to all OOV words to -unk tag and add them to vocabulary and accordingly calculate probability

- addsmooth 0 is no smoothing techniques
- to fix inf probabilityissue , we can use -unk tag or any smoothing techniques (addsmooth n , good -tuning smoothing, kneser-ney smoothing .

smoothing techniques	**ppl(test set)**
addsmooth1	4193.46
good -tuning smoothing	3025.88
kneser-ney smoothing	4038.69*

good tuning is best smoothing technique

2 Use a different smoothing algorithm other than -addsmooth that has been discussed in class during Step 2 and observe the perplexity score and out of vocabulary (OOV) probabilities?

Step 1: Generate n-gram count file from a text corpus

Step 2: Train a *language model* from n-gram count file obtained in Step 1 using smoothing and back-off probabilities

Step 3: Using the trained language model obtained in Step 2, estimate the perplexity of the test data and per-word log probabilities (likelihood) of the word sequences

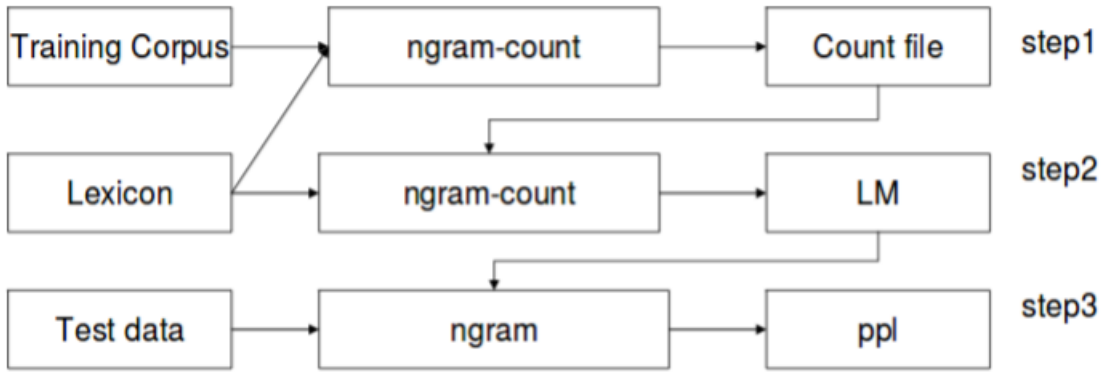

Figure 1: *n*-gram language modeling

- good -tuning smoothing
- kneser-ney smoothing

are two smoothing techniques other than addsmooth.

3 Analyze the output of your LM model by varying n (say n = 3), i.e., consider training and testing of a trigram LM model?

order	ppl
1	912.033
2	4549.2
3	4457.31
4	4457.31
5	4457.31

hence order 1 has minimum perplexity so good model.

4 How does your model prediction get affected if you pre-process your training and testing data with operations viz., punctuation removal or use of STOP words?

Preprocessing

```
[62] #preprocessing
     !cat browndata/brown-train.txt | tr '[:upper:]' '[:lower:]' | sed -e "s/[[:punct:]]\+//g" > browndata/brown-train1.txt
```

perplexity without -preprocessing = 6270.9
perplexity with -preprocessing = 4549.2
hence with preprocessing model give better result.

5 Can you train an interpolated model such as combination of bigram and trigram language models? [Hint : Use -mix-lm and -lamda

5.1 Train a separate unigram and bigram LM model with small values of -addsmooth

```
#Training bigram model on the train set
!./ngram-count -text browndata/brown-train1.txt -order 1 -write brown-train.count -unk -lm brown-train-pre-o1.lm -addsmooth 1

!./ngram-count -text browndata/brown-train1.txt -order 2 -write brown-train.count -unk -lm brown-train-pre-o2.lm -addsmooth 1
```

5.2 Jointly use both these trained models obtained above to test on your development set brown-dev.txt as indicated in Step 3 of Figure 1. One can use -mix-lm argument and -lamda weight parameter for the same.

```
!./ngram -lm /content/brown-train-pre-o2.lm -mix-lm /content/brown-train-pre-o1.lm \
-lambda 0.1 -ppl browndata/brown-dev.txt
```

5.3 Find out best lamba [0,1] (also known as interpolation weight) and -addsmooth parameters for which you get less perplexity on your development set. This process is also known as hyperparameter tuning.

lambda	ppl
0.1	713.237
0.2	641.194
0.3	602.486
0.4	581.34
0.5	572.765
0.6	575.865
0.7	593.174
0.8	633.951
0.9	734.667

addsmooth (lambda = 0.5)	ppl
1	3570.67
2	3824.31
3	4038.69
4	3935.42
5	3886.21

hence lambda = 0.5 and addsmooth 1 has minimum perplexity so good model , after all hyperparameter tuning .

5.4 Finally, use the optimum values of -addsmooth and -lamda parameter obtained during hyperparameter tuning to test on the unseen dataset — brown-test.txt

```
[220] #Training bigram model on the train set
      !./ngram-count -text browndata/brown-train1.txt -order 1 -write brown-train.count -unk -lm brown-train-pre-o1.lm -addsmooth 1

      !./ngram-count -text browndata/brown-train1.txt -order 2 -write brown-train.count -unk -lm brown-train-pre-o2.lm -addsmooth 1

      #Testing on dev set
      !./ngram -lm /content/brown-train-pre-o2.lm -mix-lm /content/brown-train-pre-o1.lm \
      -lambda 0.5 -ppl browndata/brown-dev.txt
```

```
/content/brown-train-pre-o2.lm: line 939: warning: non-zero probability for <unk> in closed-vocabulary LM
/content/brown-train-pre-o1.lm: line 938: warning: non-zero probability for <unk> in closed-vocabulary LM
file browndata/brown-dev.txt: 5734 sentences, 126831 words, 0 OOVs
0 zeroprobs, logprob= -470970 ppl= 3570.67 ppl1= 5168.55
```

```
[221] #Testing on test set
      !./ngram -lm /content/brown-train-pre-o2.lm -mix-lm /content/brown-train-pre-o1.lm \
      -lambda 0.5 -ppl browndata/brown-test.txt
```

```
/content/brown-train-pre-o2.lm: line 939: warning: non-zero probability for <unk> in closed-vocabulary LM
/content/brown-train-pre-o1.lm: line 938: warning: non-zero probability for <unk> in closed-vocabulary LM
file browndata/brown-test.txt: 14334 sentences, 305056 words, 0 OOVs
0 zeroprobs, logprob= -1.15701e+06 ppl= 4193.46 ppl1= 6205.69
```