Text Mining

MP1_Part2: Language Model

Prof. Hongning Wang

Lin Gong (lg5bt)

42

2.1 Maximum likelihood estimation for statistical language models with proper smoothing

2.1.1 Implementation of two bigram language models.

Implementation of linear interpolation smoothing.

```
//Linear Interpolation smoothing.
public double calcLinearSmoothedProb(Token token) {
    double prob = 0;
    String[] unigrams = token.getToken().split("-");
    if (m_N > 1) {
        if (m_model.containsKey(token.getToken())) {
            double a = m_stats.get(unigrams[0]).getTTF();
            double b = m_reference.m_model.get(unigrams[1]).getValue();
            prob = m_lambda * token.getTTF() / m_stats.get(unigrams[0]).getTTF() + (1 - m_lambda) * m_reference.m_model.get(unigrams[1]).getValue();
        } else {
            if(m_reference.m_model.containsKey(unigrams[1])){
                prob = (1 - m_lambda) * m_reference.m_model.get(unigrams[1]).getValue();
            } else{
                prob = (1 - m_lambda) * 1.0 / (m_totalCount + m_delta * m_testVoca);
            }
        }
    }
    return prob;
}
```

how did you estimate this?

does this mean $\delta = 0.1$? we asked you to set it to 0.1?

Implementation of absolute discount smoothing.

```
//Absolute discount smoothing.
public double calcAbsoluteSmoothedProb(Token token){
    double prob = 0, lambda = 0, S =0;
    String[] unigrams = token.getToken().split("-");
    if(m_S.containsKey(unigrams[0])){
        S = m_S.get(unigrams[0]).size();
    } else S = 0;
    if( S == 0 && m_reference.m_model.get(unigrams[0]).getTTF()==0){
        lambda = 0;
    } else {
        lambda = m_delta * S / m_reference.m_model.get(unigrams[0]).getTTF(); //The lambda used in equation.
    }
    if (m_N > 1){
        if(m_model.containsKey(token.getToken())){
            double max = (token.getTTF()- m_delta) > 0 ? (token.getTTF()- m_delta) : 0;
            prob = max / m_stats.get(unigrams[0]).getTTF() + lambda * m_reference.m_model.get(unigrams[1]).getValue();
        } else{
            if(m_reference.m_model.containsKey(unigrams[1])){
                prob = lambda * m_reference.m_model.get(unigrams[1]).getValue();
            } else{
                prob = lambda * 1.0 / (m_totalCount + m_delta * m_testVoca);
            }
        }
    }
    return prob;
}
```

→ should be 1.0

−1

According to the slides and my experiments, I find the S in equation should be all types of words occurring after a given word rather than the whole vocabulary size. If I use the vocabulary size, then the probability of \sum_P(w_i|w_i-1) is not equal to one. I also give the proof below:

thanks, you are right!

Absolute Discounting:

$$p(w_i \mid w_{i-1}, \cdots w_{i-n+1}) = \frac{\max(c(w_i, w_{i-1}, \cdots w_{i-n+1}) - \delta, 0)}{c(w_{i-1}, \cdots w_{i-n+1})} + \lambda \bar{p}(w_i \mid w_{i-1}, \cdots w_{i-n+2})$$

$$\lambda = \frac{\delta S}{c(w_{i-1}, \cdots w_{i-n+1})}$$

As we know. $\sum_{w_i} p(w_i \mid w_{i-1}, \cdots w_{i-n+1}) = 1$, ($w_i$: seen words + unseen words).

Therefore, we can get:

$$\sum_{w_i} p(w_i \mid w_{i-1}, \cdots w_{i-n+1}) = \overset{①}{\boxed{\sum_{w_i} \frac{\max(c(w_i, w_{i-1}, \cdots w_{i-n+1}) - \delta, 0)}{c(w_{i-1}, \cdots w_{i-n+1})}}} + \overset{②}{\boxed{\sum_{w_i} \frac{\delta S}{c(w_{i-1}, \cdots w_{i-n+1})} \bar{p}(w_i \mid \cdots)}}$$

{ for seen words in w_i, $\max(c(w_i, w_{i-1}, \cdots w_{i-n+1}) - \delta, 0) = c(w_i, w_{i-1} \cdots) - \delta$.
for unseen words in w_i, $\max(\cdots\cdots) = 0$.

If we assume there are k seen words, then we minus δ k times.

Rewrite part ① as:

$$\underbrace{\sum_{w_i} \frac{c(w_i, w_{i-1}, \cdots w_{i-n+1})}{c(w_{i-1} \cdots w_{i-n+1})}}_{1} + \frac{-k\delta}{c(w_{i-1}, \cdots w_{i-n+1})}$$

Rewrite part ② as:

$$\frac{\delta S}{c(w_{i-1}, \cdots w_{i-n+1})} \cdot \underbrace{\sum_{w_i} \bar{p}(w_i \mid w_{i-1} \cdots)}_{1}$$

Since ① + ② = 1, $\cancel{1} - \frac{k\delta}{c(\cdot)} + \frac{\delta S}{c(\cdot)} = \cancel{1}$

$\therefore k = S$.

Therefore, k is the $\underbrace{\text{words}}_{\text{possible number of words}}$ that following w_i.

good exploration!!, +2

2.1.2 Top ten words following "good".

Linear interpolation

```
good-eo 0.34524085740673754
good-and    0.05010133909318111
good-food   0.03186822966834152
good-as 0.027009920446559355
good-but    0.026674797534351265
good-for    0.015862224660455344
good-thing  0.014018372379815121
good-place  0.01349218298600822
good-servic 0.013450408309584631
good-too    0.012231227555345938
0.9990103672618639
```

Absolute discount

```
good-eo 0.37149468215972925
good-and    0.05246466115743195
good-food   0.034807923350750905
good-as 0.029610958162301183
good-but    0.02878405470477222
good-for    0.016526845641099585
good-thing  0.01545387504906812
good-servic 0.014674555410421215
good-place  0.014399242189518748
good-too    0.013409217769159092
0.9989004080935491
```

Any explanation of why?

(-1)

Because most of the top bigrams are seen.

2.1.3 Observation about the top words.

As we can see, the top 10 words from the two bigram language models are the same. The order and probability are slightly different. By the way, I use eos to represent the punctuations, so the good-eo means *good-punctuation*. Also, the final line is the total sum of all the words following good, which sums up to one.

2.2 Generate text documents from a language model

(-3)

2.2.1 Implementation of sampling procedure from a language model.

For unigram, I sample every unigram according to its probability.

```
//Sample the document according to unigram probability.
public void sampling() {
    double total = 0, start = 0, end = 0;
    for (Token t : m_model.values()) {
        total += t.getValue();
    }
    for (Token t : m_model.values()) {
        start = end;
        end = t.getValue() / total + start;
        m_samples.add(new Sample(start, end, t.getToken(), t.getValue()));
    }
}
```

how about smoothing?

I guess t.getValue() isn't smoothed.

For bigram, I find all the tokens following w_i-1 and sample among all the found tokens.

```
//Sample the documents according to bigram value.
public ArrayList<Sample> samplingArray(ArrayList<Token> tokens){
    ArrayList<Sample> samples = new ArrayList<Sample>();
    double total = 0, start = 0, end = 0;
    for(Token t: tokens){
        total += t.getValue();
    }
    for(Token t: tokens){
        start = end;
        end = t.getValue() / total + start;
        samples.add(new Sample(start, end, t.getToken(), t.getValue()));
    }
    return samples;
}
```

a unique way to implement this sampling!

2.2.2 The 10 sentences generated from unigram, bigram Language models.

The 10 sentences generated from unigram.

```
Log likelihood: -248.97644515348532
kitcheo wonderfulll accoutr glazeeosmarinad sjceoseoseosbundl ep breasteoseoseoseosbut menueosfolk war menuseoseosunprofession atmeosvisa e
osumeoseoseo mealeosarriv replenish eosu
Log likelihood: -227.67569941644868
ep bf ep motherfreak bf war ep trl coffeecak og atmeosvisa menueoseoseosalway ep betweeneoseoseosum pattycak
Log likelihood: -250.103750380104
ouu ep bajadera caq ingredientseosthi ep bess oo chaireoscouch war wf eossidenoteeo varrick eosloudeoseo briocheeosstyl
Log likelihood: -248.85107390602795
eventseosthank taxnot ep ep glazeeosmarinad cinnamoneoscolaeostequila kilroy atmeosvisa menuseoseosunprofession hoteleospenn ep manaeosish
at menuseoseosunprofession reservationeosfantast
Log likelihood: -237.26280811388722
ubereoscheap weareoseosteosshirt byeostheeosbottl featherweight ep kobbler ep eosr +numoveral sjceoseoseosbundl untouch thar reservationeos
fantast bus stareoseoseosjust
Log likelihood: -248.83749856657315
bō hac oileveryth sweeteospotato delivereoseosw hippo butteosrock break gersthaus vindalooeoschicken trl wf ep lai onh
Log likelihood: -262.53673338664805
myselfeosp oo cq basementeoshowev friendeosthre menuseoseosunprofession eoscloistereoshoney dinereosdivers taxnot gey og bajadera atmeosvis
a atmeosvisa helpfu
Log likelihood: -230.03063708638916
ep b dinereosdivers basementeoshowev ceosw it babysittereosc basementeoshowev og boxi ep jelloeoseoseoseo ep dinereosdivers hav
Log likelihood: -263.5979011615641
epinard atmeosvisa bajadera sjceoseoseosbundl dinereosdivers khanna serviceeostruli werd eossidenoteeo ep highlyl bō pictureeosmenu moot ba
sementeoshowev
Log likelihood: -265.7635773811654
hac eoshhhhmmmm checkeosgovea sternum eoseoseosmmmmmeoseoseoseoslick taxnot huseyin bwweoseoseoseo dinereosdivers thar subeosbar og eoseose
n orbox numeoshummus
```

You are supposed to generate "sentence" rather than a sequence of "bigrams", at least perform some pose-processing on it?

-2

The 10 sentences generated from linear interpolation bigram.

```
Log likelihood: -61.11931740034923
bus bus-staff staff-and and-most most-place place-is is-that that-it it-eoseoseo eoseoseo-i i-came came-here here-is is-veri veri-good
Log likelihood: -82.443127406751
cupeosplateeosbowl cupeosplateeosbowl-set set-eo eo-go go-wrong wrong-eo eo-hangout hangout-pace pace-mannereostim mannereostim-somehow som
ehow-still still-saliv saliv-here here-eo eo-eo
Log likelihood: -60.34069418199775
eossmedium eossmedium-oxtail oxtail-soup soup-eo eo-eo eo-my my-fianc fianc-eo eo-eo eo-the the-street street-park park-is is-made made-it
Log likelihood: -75.95887743438207
scacciata scacciata-and and-pile pile-high high-recommend recommend-it it-eo eo-and and-one one-and and-shred shred-chicken chicken-was was
-your your-choic
Log likelihood: -72.63309544491848
breasteoseoseoseosbut breasteoseoseoseosbut-now now-eo eo-i i-eosm eosm-impress impress-is is-tasti tasti-eo eo-not not-so so-eo eo-the the
-regist regist-our
Log likelihood: -73.59484069177704
basementeoshowev basementeoshowev-eo eo-known known-as as-well well-eo eo-numeo numeo-eo eo-but but-if if-you you-eo eo-eo eo-and and-carol
ina
Log likelihood: -83.1009840018653
gp gp-eo eo-ask ask-tourist tourist-guideeoscoupon guideeoscoupon-book book-eo eo-the the-beach beach-eo eo-also also-trendi trendi-and and
-you you-eosr
Log likelihood: -75.4659862167693
breadseoseo breadseoseo-all all-the the-best best-salad salad-consist consist-i i-got got-the the-wall wall-look look-pack pack-eo eo-i i-h
ad
Log likelihood: -81.22244384037297
mold mold-it it-up up-as as-the the-bar bar-new new-salon salon-eo eo-i i-had had-at at-stick stick-to to-order
Log likelihood: -70.38408610157475
bajadera bajadera-hazelnut hazelnut-cake cake-all all-of of-a a-bonus bonus-for for-numeosnum numeosnum-star star-eo eo-eo eo-when when-we
we-sat
```

The 10 sentences generated from absolute discount bigram.

```
Log likelihood: -73.13835986113139
animaleoslov animaleoslov-vegan vegan-top top-with with-shred shred-of of-the the-hummus hummus-eo eo-but but-then then-it it-eo eo-the the
-valu
Log likelihood: -72.02770943966333
eoseoscmon eoseoscmon-now now-got got-hard hard-to to-die die-to to-get get-to to-order order-was was-fresh fresh-ingredi ingredi-eo eo-i
Log likelihood: -68.51445699865522
refillseoseosnum refillseoseosnum-star star-eo eo-the the-plate plate-eo eo-pack pack-eo eo-it it-is is-that that-made made-a a-decis decis
-eo
Log likelihood: -65.0050740821315
selyodka selyodka-eo eo-and and-i i-eosm eosm-tri tri-to to-help help-it it-eoss eoss-one one-night night-we we-decid decid-to
Log likelihood: -73.30276768454875
ep ep-eo eo-make make-the the-food food-at at-onc onc-way way-off off-eo eo-it it-also also-love love-this this-place
Log likelihood: -76.7577485918354
ep ep-eo eo-torta torta-eo eo-and and-delish delish-though though-i i-love love-the the-littl littl-did did-neost neost-move move-eo
Log likelihood: -88.21450886782581
babysittereosc babysittereosc-this this-was was-lunch lunch-hour hour-eo eo-ice ice-cream cream-and and-miss miss-the the-seaport seaport-a
nd and-ate ate-here
Log likelihood: -78.89590759524977
atmeosvisa atmeosvisa-card card-eo eo-excel excel-choic choic-eo eo-beer beer-and and-diet diet-coke coke-and and-airi airi-room room-in in
-vega
Log likelihood: -75.12524744603333
burritophil burritophil-eo eo-our our-last last-time time-eo eo-this this-is is-their their-dinner dinner-tonight tonight-with with-their t
heir-hous hous-and
Log likelihood: -81.71151877720534
fruityback fruityback-porch porch-last last-night night-eo eo-no no-differ differ-but but-i i-drove drove-eo eo-eo eo-casual casual-come co
me-on
```

2.3.1 Implementation of the perplexity calculation of a language model.

For unigram Language Model:

```java
//Calculat the unigram perplexity for one doc.
public double calcUnigramDoc(Doc d, int V){
    double prob = 1;
    String[] tokens = d.getTokens();
    for(int i = 0; i < tokens.length; i++){
        if(m_model.containsKey(tokens[i])){
            prob = prob * Math.pow((m_model.get(tokens[i]).getTTF()+m_delta) / (m_totalCount + m_delta * V), 1.0 / tokens.length);
        } else{
            prob = prob * Math.pow(1.0 / (m_totalCount + m_delta * V), 1.0 / tokens.length);
        }
        if(prob == 0)
            break;
    }
    return 1.0 / prob;
}
```

For bigram Language Model:

-3

```
//Calculate the bigram perplexity for one doc.
public double calcBigramDoc(Doc d, String method){
    double prob = 1;
    String[] tokens = d.getTokens();
    for(int i = 0; i < tokens.length - 1; i++){
        String bigram = tokens[i] + "-" + tokens[i+1];
        Token t = new Token(bigram);
        if(method.equals("LI")){
            prob = prob * Math.pow(calcLinearSmoothedProb(t), 1.0 / tokens.length);
        }else if(method.equals("AD")){
            prob = prob * Math.pow(calcAbsoluteSmoothedProb(t), 1.0 / tokens.length);
        }
    }
    double value = 1.0 / prob;
    return value;
}
```

2.3.2 Report the mean and standard deviation of three language models.

	Average	Standard Deviation
Unigram LM	2938.6897819864757	1.7999978448758727E8
Linear Interpolation LM	4046.2693776047086	1258265.3000934576
Absolute Discount LM	Infinity	NaN

2.3.3

According to the experimental results, unigram performs best among all the three languages.

In Absolute Discount LM, the reason for infinity is that the probability of a word is too small to be caught by machine. So the values are replaced with zero. Since we have smoothed all unseen words, there are no unseen words.

does this follow our expectation?