CS 6501: Text Mining

Homework 1: MP1_Part 2

Student: Mohammad Al Boni

NetID: ma2sm

Pledge

Part 2.1:

1. Smoothing code:

```
        public double calcLinearSmoothedProb(String token) {
                if (m_N>1) // To make this condition work for m_N>1, we need to have an array of lambdas
that sums up to one. Since in the assignment we have only bigrams, I will keep lambda as a scalar.
                   return m_lambda  * calcMLProb(token) +  (1.0-m_lambda)*
                          m_reference.calcLinearSmoothedProb(token.split("-")[1]) ;
                else
                   return  ((m_model.containsKey(token)? m_model.get(token).getValue():0)+ 0.1)/(double)(
                             TotalNumberOfWords+0.1*m_model.size());
                }
public double calcAbsoluteDiscountSmoothedProb(String token) {
                if (m_N>1)
                return  (Math.max((m_model.containsKey(token)? m_model.get(token).getValue():0)-m_delta,0)
                        +m_delta*m_S.get(token.split("-")[0])*
                         m_reference.calcAbsoluteDiscountSmoothedProb(token.split("-")[1]))/
                        (double)m_reference.m_model.get(token.split("-")[0]).getValue();
                else
                return  ((m_model.containsKey(token)? m_model.get(token).getValue():0)+ 0.1)/(double)(
                          TotalNumberOfWords+0.1*m_model.size());
                }
```

Note, to make sure that the smoothed probabilities sum to one, I chose to ignore the sentence boundaries. I virtually connected all sentences and documents so that if the corpus contains N unigrams, it will contain N-1 bigrams. For example, if we have the following document: "I like this place. The service is very good!". After ignoring the sentence boundaries, "place-the" will be added to the bigrams. This approach will introduce some noise to the bigram probabilities but will guarantee that all probabilities sum to one.

2. Top words that follow "good":

Linear interpolation:

Rank	Word	Smoothed Likelihood	Rank	Word	Smoothed Likelihood
1	and	0.066786829	6	as	0.029125288
2	but	0.063108499	7	for	0.018319332
3	i	0.057450023	8	it	0.017351858
4	the	0.051914019	9	servic	0.016824377
5	food	0.033408324	10	too	0.014714916

Absolute discount:

Rank	Word	Smoothed Likelihood	Rank	Word	Smoothed Likelihood
1	and	0.070595923	6	as	0.031908656
2	but	0.069149206	7	for	0.019114119
3	i	0.060594959	8	servic	0.01838045
4	the	0.052047449	9	it	0.01740592
5	food	0.036437114	10	too	0.016140714

3. Observations:

Theoretically, the main purpose of smoothing to eliminate the case of having zero probabilities where the unseen words will gain some value taken from the seen ones. Therefore, regardless of the smoothing method used, the rank of the n-grams, which is based on the maximum likelihood estimate, should not change. This concept is

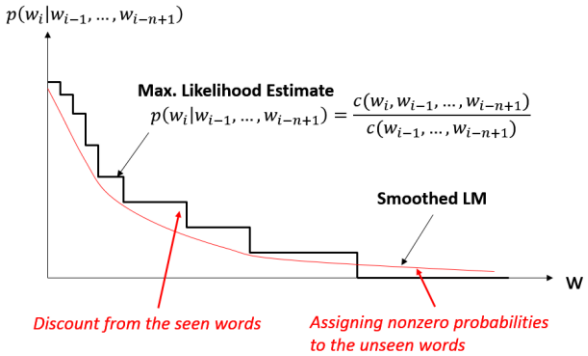

Figure 1. Maximum likelihood LM vs. smoothed LM.

illustrated in Fig. 1. For example, let's have an unsmoothed LM, and let "good-food" be a bigram with rank 2 and maximum likelihood of 0.3, then regardless of the smoothing method, "good-food" will exactly have rank 2 but the new probability estimate will be less than 0.3 which depends on the smoothing technique.

Technically, the linear interpolation smoothing method is defined as:

$$\bar{p}(w_i|w_{i-1},\dots,w_{i-n+1}) = \lambda p_{ML}(w_i|w_{i-1},\dots,w_{i-n+1}) + (1-\lambda)\bar{p}(w_i|w_{i-1},\dots,w_{i-n+2})$$

while the absolute discount method is defined as:

$$\bar{p}(w_i|w_{i-1},\dots,w_{i-n+1}) = \frac{\max\left(c(w_i,w_{i-1},\ \dots,w_{i-n+1})-\delta,0\right)+\delta S\bar{p}(w_i|w_{i-1},\dots,w_{i-n+2})}{c(w_{i-1},\ \dots,w_{i-n+1})}$$

Note, both methods depend on the counts of the n-grams. Therefore, if both models were estimate from the same corpus, then the rank will remain the same. However, the difference between the two methods is that in linear interpolation (LI) we discount by a percentage of the maximum likelihood probability. For example, if λ is 0.9, then 10% of the ML will be assigned to the unseen words. As a result, n-grams with higher ML will be discounted more than n-grams with lower ML. On contrast, in absolute discount (AD), all n-grams will be discounted by a fixed value, δ, regardless of their own ML probability. This explains why the probabilities of the top 10 words using AD methods are higher than the ones using the LI method because the top words in LI will be discounted more than other n-grams. This is also the reason ranks of "servic" and "it" are swapped in the two different methods. Finally, different λ and δ parameters will change the behavior of these smoothing methods. For example, using $\lambda = 0.95$, we get the following ranking:

Rank	Word	Smoothed Likelihood	Rank	Word	Smoothed Likelihood
1	and	0.068656388	6	as	0.03051268
2	but	0.066119521	7	for	0.018704777
3	i	0.05899113	8	servic	0.017599477
4	the	0.051926087	9	it	0.017360787
5	food	0.034916187	10	too	0.01542589

Note that probabilities are higher that the case when $\lambda = 0.95$, and "servic" and "it" have the same rank as for when AD was used. To better understand the effect of using different parameters, I ran several tests using various values for λ and δ. Figures 2 and 3 shows the smoothed probabilities of all possible 385432 words that follows "good" using LI and AD smoothing methods respectively. In LI, λ was set to 0.25, 0.50, 0.75, 0.90 and 0.99. As I mentioned above, the higher the λ is, the less percentage of ML is discounted from the seen bigrams which is assigned to the unseen ones. When λ=0.99, we apply almost no smoothing and the probability of the unseen words is almost equal to zero. Figure 4 shows a clearer difference between the very low smoothing (λ=0.99) and high smoothing (λ=0.25). On the other hand, Figure 3 shows the effect of changing δ on the behavior of the

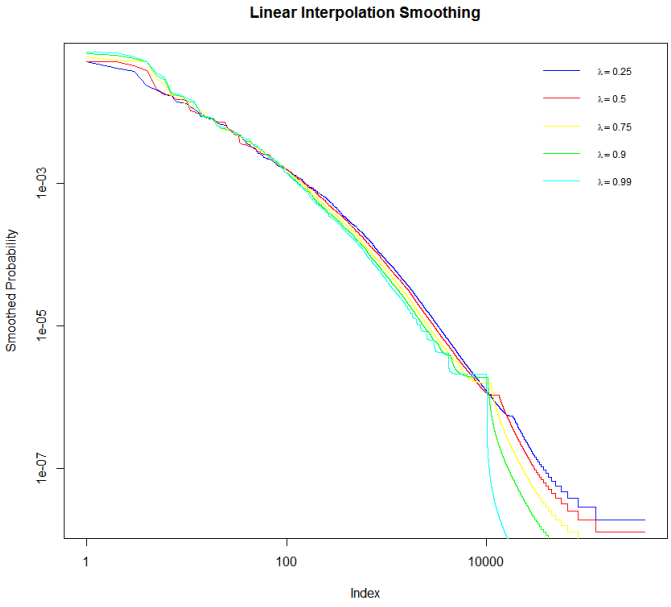

Figure 2. Linear interpolation smoothing

smoothing when AD methods is applied. Note that since the discounting is fixed rather than being a percentage, all seen bigrams are discounting with the same δ. As a result, regardless of δ, the shape of the gram for the seen bigrams will be the same. An extremely important property of AD methods is also being observed in Figures 3 and 5: the area under the curve for the unseen bigrams is distributed differently when various δs are used. When δ is low (0.25 for example), more probability is assigned to the bigrams which has higher unigram probabilities for their seen part. This property is caused by multiplying δS by $\bar{p}(w_i|w_{i-1},\dots,w_{i-n+2})$. Figure 5 clearly shows this property when only curves with δ=0.25 and δ=0.99 are compared.

A simple answer is that most top bigrams are seen.

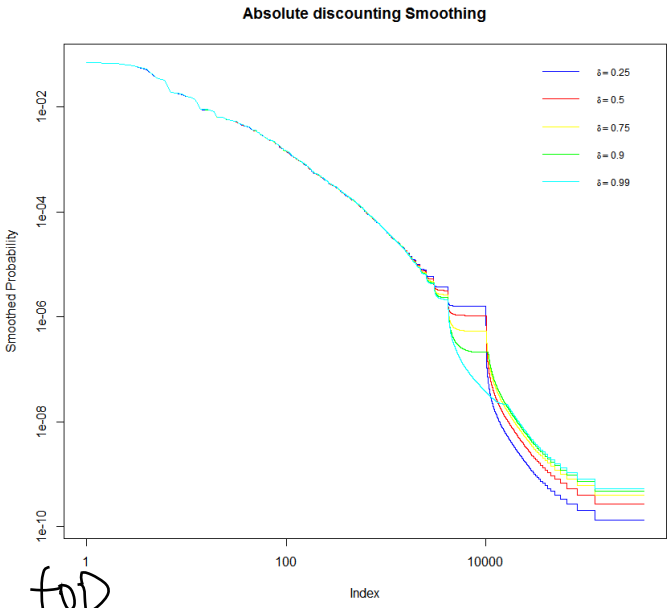

Figure 3. Absolute discounting smoothing

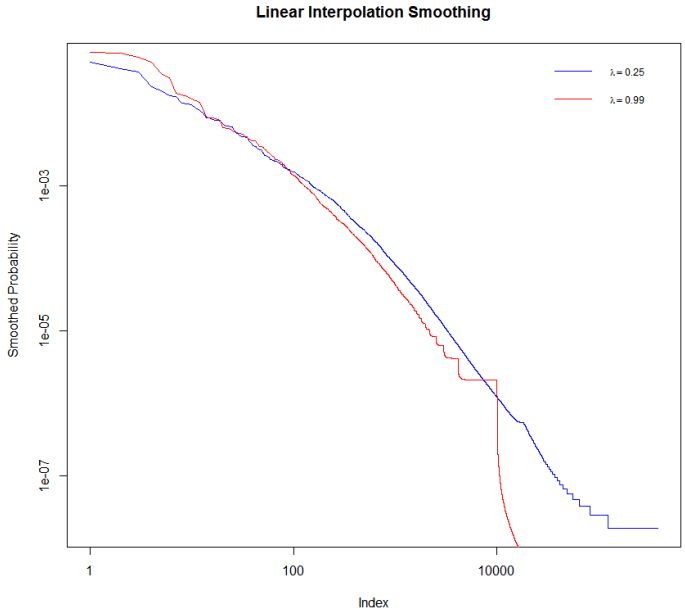

Figure 4. Linear interpolation smoothing

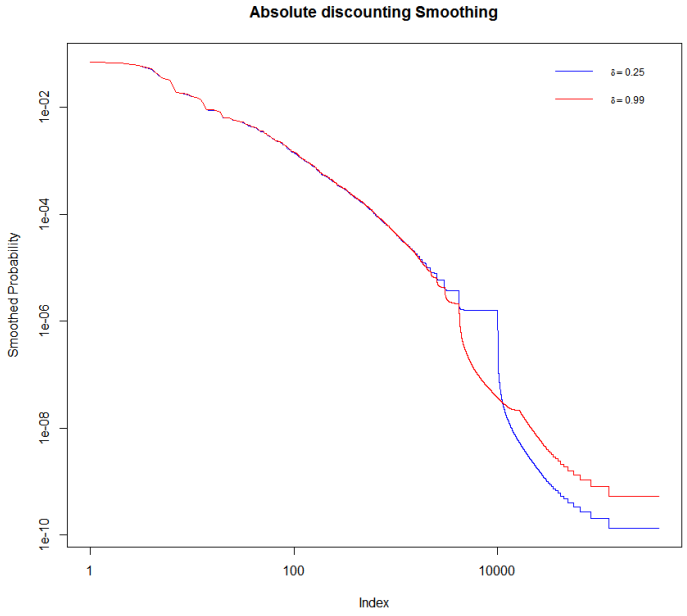

Figure 5. Absolute discounting smoothing

Very good exploration!

+5

Part 2.2:

1. Sampling code:

```
public String SampleFromUnigram(){
        double RandomNumber=R.nextDouble();
        double Sum=0;
        for(Token t:m_unigramLM.m_model.values())
        {
                Sum+=m_unigramLM.calcLinearSmoothedProb(t.getToken());
                if(Sum>=RandomNumber)
                        return t.getToken();
        }
        return "";
}
public String SampleFromBigram(String ObservedToken,SmoothingMethod Method){
        double RandomNumber=R.nextDouble();
        double Sum=0;
        for(Token t:m_unigramLM.m_model.values())
        {
             Sum+=Method==SmoothingMethod.LinearInterpolation?
             m_bigramLM.calcLinearSmoothedProb(ObservedToken+"-"+t.getToken()):
             m_bigramLM.calcAbsoluteDiscountSmoothedProb(ObservedToken+"-"+t.getToken());
            if(Sum>=RandomNumber)
                return t.getToken() ;

        }
        return "";
}
```

2. Generated Sentences:
    a. Unigram:

the first of sandwich my it for and like pud nt my were when time
is littl can would so select of been where go time ramen is in to
tricolor oh tougher our did and dure s out and the nonchal here love casino
roy not wheat the group would all to those got lot regist they chewi if
this for dine NUM their an of they would great price again to chees staff
plate i suggest for clear but NUM or this whole away veri m i and
hoop some could a on the packag bad followingfrutti as this with also were restaur
what same on have and out a also order and a coupl happen omg it
egg a to the week such and placefor use and s that i appet a
but the readingat across no do down glass spot sandwich like the oyster of i

    b. Bigram (Linear interpolation):

we did nt go here my list was safe our food the back after not
coupon help i stop eat out websit which crispi exterior fool you go to eat
of time but well i felt like my tast to us for that in the
better chicken and through stl and chalupa quesadilla and made fresh and a special to
NUM if you d mention the same as comfort all you can enjoy my salon
mole enchilada they do anyth about thenthuk tradit japanesey i use to do nt mind
stool with with perfect saute kale yuk i live in flavor as well prepar and
the bar around to was class and have receiv my salad am NUMstar but drop
saturday night but the and prepar and my plan for eat in the onli NUM
it cook realli good food was nt too crowd is nt cut back again the

c. Bigram (Absolute discount):

staff beyond the servic was alway fresh dish was mushroom saute spinach day i had
enorm curtain when i m take your here with a few kink my trip and
but it was obvious this place is calm zen was happi with a martini which
of water and drum is a bottl of we were delici beauti as soon like
damn this at one of lighter so much of the spot the theater dinner in
measur with some other would be a verit factori so we flag over his masterpiec
bagel puff pastri while she made with me a tomato in the lamb chop salad
salt and gourmet pizza both were heaven amaz i guess what it was not use
enthusiasm we consum of five i ll complain becaus i love it i love ice
my favorit wingstop anoth aspect great servic was realli awesom it was fine with a

Part 2.3:

1. Perplexity calculation code:

```
public void analyzeDocumentDemo(JSONObject json,int index,int core ) {
		try {
			JSONArray jarray = json.getJSONArray("Reviews");
			for(int i=0; i<jarray.length(); i++) {
				Post review = new Post(jarray.getJSONObject(i));
				double uni=1;
				double biLI=1;
				double biAD=1;
				int WordsCount=0;
				// get actual length of each review
				ArrayList<String> processedTokens=new ArrayList<String>();
				for(String token:tokenizer.get(core).tokenize(review.getContent())){
					String finalToken=SnowballStemmingDemo(NormalizationDemo(token));
					// if the token is empty, then try next token
					if(!finalToken.isEmpty()){
						WordsCount++;
						processedTokens.add(finalToken);
					}
				}
				String previousToken="";
				for(String finalToken:processedTokens){
						//unigram

				 uni*=Math.pow(1/m_unigramLM.calcLinearSmoothedProb(finalToken),
							1/(double)WordsCount);
						 // bigram
						 if(!previousToken.isEmpty()){
						 String vocabID=previousToken+"-"+finalToken;
				 biLI*=Math.pow(1/m_bigramLM.calcLinearSmoothedProb(vocabID),
						 1/(double)WordsCount);
				 biAD*=Math.pow(1/m_bigramLM.calcAbsoluteDiscountSmoothedProb(vocabID),
						 1/(double)WordsCount);
						 }
						 previousToken=finalToken;
				}
				synchronized(lock1) {
					UniPP.get(index).add(uni);
					BiLIPP.get(index).add(biLI);
					BiADPP.get(index).add(biAD);
				}
			}
		} catch (JSONException e) {
			e.printStackTrace();
		}
		}
```

2. Mean, median and standard deviation:

Table 1. Perplexities of unigram and two bigram LMs on the test data.

	Unigram LM	Bigram LM (linear)	Bigram LM (Absolute)
Mean	28845.08	757.3724	617.6462
Standard Deviation	4618475	184222.9	111583.5
Min	1	1	1
25 Percentile	466.1084	102.1197	101.4061
Median	591.5589	143.271	147.0169
75 Percentile	784.6190	209.6902	224.5742
Max	786732142	76704663	46290424

3. Discussion and conclusions:

Two important things regarding the implementation need to be mentioned. First, the absolute discounting smoothing method need to be adjusted to account for the case where w_{i-1} is unseen. The original formula is

$$\bar{p}(w_i|w_{i-1}) = \frac{\max(c(w_i, w_{i-1}) - \delta, 0) + \delta S \bar{p}(w_i)}{c(w_{i-1})}$$

Since w_{i-1} is unseen, then $c(w_{i-1}) = 0$ and $c(w_i, w_{i-1}) = 0$. As a result, $\frac{\max(c(w_i,w_{i-1})-\delta,0)}{c(w_{i-1})} = \frac{0}{0}$. We will set $\frac{0}{0} = 0$. Furthermore, in order to make all probabilities sum to 1 (i.e., $\forall v_j \in V, \sum_i^n \bar{p}(w_i = v_i | w_{i-1} = v_j) = 1$ where V is the set if all vocabulary), then we need to set λ to 1 instead of $\frac{\delta S}{c(w_{i-1})}$. The final AD equation in this case will be:

$$\bar{p}(w_i|w_{i-1}) = \frac{\max(c(w_i,w_{i-1})-\delta,0)+\delta S\bar{p}(w_i)}{c(w_{i-1})} = \frac{\max(c(w_i,w_{i-1})-\delta,0)}{c(w_{i-1})} + \frac{\delta S\bar{p}(w_i)}{c(w_{i-1})} = 0 + \lambda\bar{p}(w_i) = \bar{p}(w_i).$$

This is proved using the conditional probability formula:

$$\bar{p}(w_i|w_{i-1}) = \frac{\bar{p}(w_i \cap w_{i-1})}{\bar{p}(w_{i-1})}$$

Since $w_{i-1}$is unseen, then w_i and $w_{i-1}$are independent, i.e., $\bar{p}(w_i \cap w_{i-1}) = \bar{p}(w_i) * \bar{p}(w_{i-1})$. Therefore,

$$\bar{p}(w_i|w_{i-1}) = \frac{\bar{p}(w_i) * \bar{p}(w_{i-1})}{\bar{p}(w_{i-1})} = \bar{p}(w_i)$$

Or in another words, the smoothed probability of a bigram that is conditioned on an unseen token is equals to the smoothed probability of the unigram in hand.

The second implementation issue is the computation of the perplexity. In some cases when we have relatively large reviews that contain a lot of unseen words, the denominator will approach to zero and the perplexity for those reviews will equal to infinity. In order to overcome this issue, we can use a simple trick in which we distribute the square root into the perplexity equation to avoid the effect of multiplying with many very small numbers together. The distribution process for the case of unigram LMs:

$$PP(w_1, \dots, w_N) = \sqrt[N]{\frac{1}{\prod_{i=1}^{N} p(w_i)}} = \sqrt[N]{\frac{1}{p(w_1)} * \frac{1}{p(w_2)} * \dots * \frac{1}{p(w_N)}}$$
$$= \sqrt[N]{\frac{1}{p(w_1)}} * \sqrt[N]{\frac{1}{p(w_2)}} * \dots * \sqrt[N]{\frac{1}{p(w_N)}}$$

Likewise, in case of bigram LMs:

$$PP(w_1, \dots, w_N) = \sqrt[N]{\frac{1}{\prod_{i=1}^{N} p(w_i|w_{i-1})}} = \sqrt[N]{\frac{1}{p(w_2|w_1)} * \frac{1}{p(w_3|w_2)} * \dots * \frac{1}{p(w_N|w_{N-1})}}$$
$$= \sqrt[N]{\frac{1}{p(w_2|w_1)}} * \sqrt[N]{\frac{1}{p(w_3|w_2)}} * \dots * \sqrt[N]{\frac{1}{p(w_N|w_{N-1})}}$$

Finally, Table 1 shows some statistics on the perplexities of unigram and two bigram LMs on the test data. Note that in all cases, unigram LM preformed the least compared to the two bigram models. This applies to the mean, median, standard deviation … etc. Also, we notice that the test data contains some outliers, i.e., some documents that are written using a different set of vocabulary than the one used in the training set. This results with a very large perplexity values which heavily affected the mean and the standard deviation. In such cases, if we choose not to remove these outliers, then the median, the 25 and the 75 percentiles are better comparison metrics because they are not affected by outliers. In terms of the median, the bigram language model that is smoothed by linear interpolation method performed the best (143.271 compared to 591.5589 and 147.0169 for the unigram and bigram using AD respectively). However, as I discussed in part 2.2, absolute discounting method will redistribute a fixed area of the probability curve to the unseen ngrams. As a result, even when having a large number of unseen ngrams (in case of the outlier documents), the same area will be distributed for these ngrams. Therefore, absolute discounting method is more robust that the linear interpolation method. This is also shown in Table 1 where the max and the mean perplexity of the test data when using AD method is smaller than the values when LI is used.