1.1

CODES. 1. Implementation of linear interpolation smoothing

```
int len=unigramList.size();
bigramCount=new int[len][len];
for(int[] bigram : bigramList){
        bigramCount[bigram[0]][bigram[1]]++;   // calculate c(w_i,w_i-1)
}
bigramLM_LIS=new double[len][len];
for(int i=0;i<len;i++){
        for(int j=0;j<len;j++){
                bigramLM_LIS[i][j]=(1.0-lamda)*bigramCount[i][j]/unigramCount.get(unigramList.get(i))
                                +lamda*unigramLM[j];
        }
}
```

CODES. 2. Implementation of absolute discount smoothing

```
bigramLM_ADS=new double[len][len];
for(int i=0;i<len;i++){
        int uniqueBigram=0;
        for(int j=0;j<len;j++){
                if(bigramCount[i][j]>=1)uniqueBigram++;  // calculate |d|u
        }
        for(int j=0;j<len;j++){
                bigramLM_ADS[i][j]=(Math.max(bigramCount[i][j]-cigma,0.0)+cigma*uniqueBigram*unigramLM[j])
                                /unigramCount.get(unigramList.get(i));
        }
}
```

1.2

Absolute Discount Smoothing		
Good	but	0.097184
	and	0.064615
	i	0.057376
	the	0.053219
	as	0.036893
	food	0.0291
	it	0.019015
	for	0.017799
	thing	0.017796
	too	0.017372

Linear Interpolation Smoothing		
Good	the	0.052704
	and	0.035286
	i	0.032259
	a	0.023764
	to	0.02008
	but	0.018354
	it	0.017868
	was	0.017231
	of	0.014668
	for	0.011869

1.3 Because we do not remove stopwords, some high-frequency but meaningless words such as “and”, “the”, “it”, “for”, “of” occur at the following position of “good”. It does not make sense because some of them should not follow “good” in real sentence. The reason is that their maximum likelihood probability in unigram language model is very high, which is used as the reference language model for bigram language model smoothing. This leads to a high value of p(.|”good”). The Absolute discount smoothing (ADS)

performs better than Linear interpolation smoothing (LIS). As shown in upper table, there are more reasonable following words like "food" and "thing" in ADS than that in LIS. This is because smoothing parameter $\lambda$of LIS is 0.9, which means the LIS bigram language model is very similar to unigram language model. By comparison, the smoothing parameter δ of ADS is 0.1, which made the ADS bigram language model is slightly influenced by the reference unigram language model.

2.1

CODES. 3. Codes for Implementation of sampling procedure from a language model.

```
// len is the length of output document.
// LM_type=0: Unigram, =1: Bigram_LIS, =2: Bigram_ADS
public String generateDocument(int len,int LM_type){
        if(len<1)return "";
        StringBuilder sb=new StringBuilder();
        double randomNum=Math.random();                         // generate a random number in (0.0,1.0).
        double sum=0.0,temp=0.0; int i=0; int j=1; int pre=0;
        for(i=0;i<unigramLM.length;i++){                        // sum of probability of each word.
                sum=sum+unigramLM[i];
                if(sum>=randomNum)break;                        // when sum > randomNum, the word is chosen.
        }
        sb.append(unigramList.get(i)); sb.append(" "); pre=i;

        for(j=1;j<len;j++){
                randomNum=Math.random(); sum=0.0;
                for(i=0;i<unigramLM.length;i++){
                        if(LM_type==0)temp=unigramLM[i];
                        else if(LM_type==1)temp=bigramLM_LIS(pre,i);
                        else temp=bigramLM_ADS(pre,i);
                        sum=sum+temp;
                        if(sum>=randomNum)break;
                }
                if(i==unigramLM.length)i=unigramLM.length-1;
                sb.append(unigramList.get(i)); sb.append(" "); pre=i;
        }
        return sb.toString();
}
```

2.2

For Unigram Language Model

20-words Sentence	Log_{10}(**likelihood**)
NUM eat with off wast actual a tri they did bar was to much bar port star all was and	-52.0071
never quartino polit it NUM is room last modern wine tinfoil worth find decent chang though s dure lot an	-64.7034
watch was cut are southeast we there NUM feed abund was you time pretti the street m i a lot	-55.627
me as of to at took santana the it to be was onli had about thing leav a of amount	-47.983
my overwork need and had in of veal and sweet throughout the peke look genuin came my can i did	-56.8275
they have some classic thought a was she onion cours same lot they spread you the resembl the restaur it	-54.1018
crab howev triangular made secret comparison pizza a it and the was s go ingredi girl menu super meat that	-58.5986
vegetarian eat and of bld have yelp mean find it wonder the good select bean were i would actual happenedth	-58.688
to total eaten there and tender are was was not just do splash order appet away with ve that oyster	-53.0727

me to and beer they back use guarante bite was more clean i delici yeah sign the jalapeno not are	-55.4266

For Bigram Language Model with Linear Interpolation Smoothing (**LIS**)

20-words Sentence	Log_{10}(**likelihood**)
and eventu and the noth food of ok normal week that be and feel have pizza complaint i food well	-51.1159
to is a at would that but consid person i chicago to nt with is chantal charg communiti undeni nt	-57.0157
as visit pasta as get was curri everi the and and serv again NUM arriv bar littl design trip i	-53.8073
a fusion atmospher would dinner drove mint but dish when it bar gorgeous tast good at portillo make the back	-57.5819
neighborhood my than NUM to citi frequent smart yum their in the saturday i me want can of brais to	-54.8053
too this i down sashimi also i sampl mac card nt they parmesan new readi human that and pipe and	-58.7395
of realli the to to twice this that they do chees insult privat them for from it back in was	-49.2295
they the next bottom offer drink mayb in was a NUM is end lucki catch i onli reason insid the	-51.8654
pie give seat simpl and recent bad more prompt anyway it date with it all nice or attent truffl crowd	-58.7906
year i as vow to i with i realli noth order we at other friend the at was am papaya	-49.978

For Bigram Language Model with Absolute Discounting Smoothing (**ADS**)

20-words Sentence	Log_{10}(**likelihood**)
the same time a oh lord the food for famili as well everyth we were realli good thing on your	-37.7277
luv u shape and the rest of ingredi are pay tabl was visit a landslideeven better and french onion and	-42.1827
tasti and the one of the white had no way through featur meat and miso soup bagel potato were pretti	-40.7464
dog is a car order our waitress brought back what are ashamet uf look for it they have a glass	-39.9524
import not have no reserv too spici and drink are pretti good there a bit sweeter side the soup and	-39.8097
supers reput of a tad pricey night and they re not tell you have grown up to tri we were	-36.7913
is also discov crisp again our waitress help the pay the staff was intrigu to start off your pizza ask	-45.3779
turn out by it a nice peopl there umm yes you are great pizza about that i live up and	-39.881
like it ll onli restaur we would write a bottl the price to choos just kid butteri crust brick paella	-48.8225
a live in the floor serv it is a week in comparison to me i like it was nt take	-33.291

3.1

$$PP(d) = \sqrt[n]{\frac{1}{\prod_{i=1}^{n} p(w_i \mid w_{i-1}, w_{i-N+1})}} \quad \Rightarrow \quad \log(PP(d)) = -\frac{1}{n}\sum_{i=1}^{n} \log(p(w_i \mid w_{i-1}, w_{i-N+1}))$$

CODES. 4. Codes for the implementation of perplexity calculation of language models.

```
public void obtainReviewsPerplexity(){
    int numReviews=reviewsList.size(); List<String> review; double sqrtNum;
    double unigramPer,bigramPerLIS,bigramPerADS; int preWordIndex,wordIndex;
    reviewsPerplexity=new double[3][numReviews]; // 0 unigram, 1 bigram_LIS, 2 bigram_ADS
    for(int i=0;i<numReviews;i++){
        review=reviewsList.get(i);
        unigramPer=0.0; bigramPerLIS=0.0; bigramPerADS=0.0;
```

```
        preWordIndex=-1;
        for(String word:review){
            wordIndex=unigramIndex.get(word);
            unigramPer=unigramPer+Math.log10(unigramLM[wordIndex]);
            if(preWordIndex>=0){
                bigramPerLIS=bigramPerLIS+Math.log10(bigramLM_LIS(preWordIndex,wordIndex));
                bigramPerADS=bigramPerADS+Math.log10(bigramLM_ADS(preWordIndex,wordIndex));
            }else{
                bigramPerLIS=bigramPerLIS+Math.log10(unigramLM[wordIndex]);
                bigramPerADS=bigramPerADS+Math.log10(unigramLM[wordIndex]);
            }
            preWordIndex=wordIndex;
        }
        sqrtNum=-1.0/review.size();
        reviewsPerplexity[0][i]=Math.pow(10.0,sqrtNum*unigramPer);
        reviewsPerplexity[1][i]=Math.pow(10.0,sqrtNum*bigramPerLIS);
        reviewsPerplexity[2][i]=Math.pow(10.0,sqrtNum*bigramPerADS);
    }
}
```

3.2

Mean & Standard Deviation of Perplexity on Test Data

	Mean	Standard Deviation
Unigram LM	5734.84	737565
Bigram LM with **LIS**	5294.34	737570
Bigram LM with **ADS**	5231.13	737543

3.3 According to the upper table, the bigram language model with absolute discounting smoothing (**ADS**) performs the best in predicting the data of test folder. Because it has the minimum value of perplexity mean. If a model can predict data well, which means the likelihood probability of the data is relatively large, which suggests a small value of perplexity.

4. Bonus

Dirichlet Smoothing for Bigram language model:

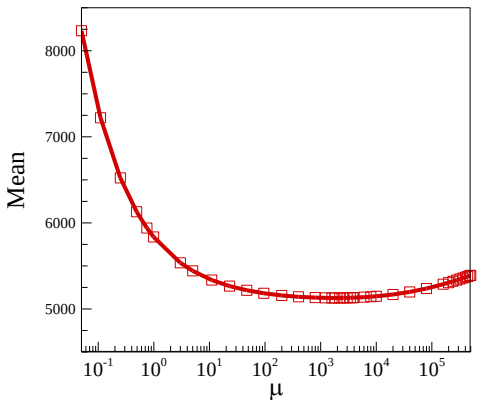
$$\hat{p}^{D}(w_i \mid w_{i-1}) = \frac{c(w_i, w_{i-1}) + \mu \hat{p}(w_i)}{|d| + \mu}$$, where $|d| = c(w_{i-1})$ in training fold. The implementation of this smoothing method is listed in CODES. 5.

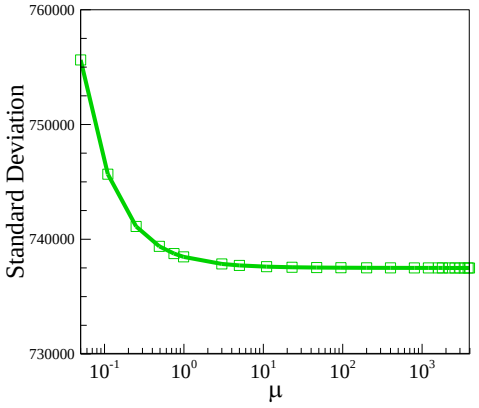
CODES. 5. Codes for implementation of perplexity calculation of Bigram language model with Dirichlet Smoothing.

```
//bigram language model with Dirishlet Smoothing
public double bigramLM_DS(int w1,int w2){
    String bigramIndex=String.valueOf(w1)+"-"+String.valueOf(w2);
    if(bigramCount.containsKey(bigramIndex)){
        return(bigramCount.get(bigramIndex)+DS_mu*unigramLM[w2])/(unigramCount.get(unigramList.get(w1))+DS_mu);
    }else{
        return DS_mu*unigramLM[w2]/(unigramCount.get(unigramList.get(w1))+DS_mu);
    }
}
//get the perplexity of reviews using bigram language model with Dirishlet Smoothing
```

```
public void obtainReviewsPerplexity_DirichletSmoothing(){
    int numReviews=reviewsList.size(); List<String> review;
    double bigramPerDS; int preWordIndex,wordIndex;
    reviewsPerplexityDS=new double[numReviews];
    for(int i=0;i<numReviews;i++){
        review=reviewsList.get(i); bigramPerDS=0.0; preWordIndex=-1;
        for(String word:review){
            wordIndex=unigramIndex.get(word);
            if(preWordIndex>=0){
                bigramPerDS=bigramPerDS+Math.log10(bigramLM_DS(preWordIndex,wordIndex));
            }else{
                bigramPerDS=bigramPerDS+Math.log10(unigramLM[wordIndex]);
            }
            preWordIndex=wordIndex;
        }
        reviewsPerplexityDS[i]=Math.pow(10.0,-1.0*bigramPerDS/review.size());
    }
}
```

A set of μ values from 0.05 to 500000 have been tried, and the evolutions of their mean and standard deviation are shown in below.





The minimum value of mean (=5128) occurs at **μ≈1800** with standard deviation=737499. We can see that this result is better than that of Absolute Discounting Smoothing ($\delta = 0.1$) because it has smaller value of perplexity mean.