Guide for Reproducibility of *Using a Text n-gram Distribution Model to Design Cache Approaches*

No Author Given

No Institute Given

1 Introduction

This is a guide to reproduce the results related to the content of the paper titled *Using a Text n-gram Distribution Model to Design Cache Approaches*. File CacheDesign.zip must be uncompressed and all files in it must be placed in the same folder. As explained the the next section, this set has python function files (*.py), which constitute the approach proposed in the paper; files containing the *spline* regressions obtained by the prediction model (as referred in the paper) during the training phase, (*.pkl); files generated after running the python functions; and files used in *gnuplot* to generate the Figures and other results of the paper. This is the whole set:

```
CacheDesignForLocalMax.py
Def_ConstantsAndGlobalVars.py
Def_DistByLangNandK.py
Def_KThresolds.py
Def_TestingCorporaSizes.py
Def_ValidationCorporaSizes.py
Def_Vocabulary.py
Def_monotony.py
Def_smooth_spline_results.pkl
Def_smooth_spline_results_const.pkl
EmpLocMaxCacheForHitRatioVsCorpusSize_en_0.75_1
EmpLocMaxCacheForHitRatioVsCorpusSize_en_0.75_2
EmpLocMaxCacheForHitRatioVsCorpusSize_en_0.85_1
EmpLocMaxCacheForHitRatioVsCorpusSize_en_0.85_2
EmpLocMaxCacheForHitRatioVsCorpusSize_en_0.95_1
EmpLocMaxCacheForHitRatioVsCorpusSize_en_0.95_2
EmpLocMaxCacheForHitRatioVsCorpusSize_en_0.99_1
EmpLocMaxCacheForHitRatioVsCorpusSize_en_0.99_2
EmpLocMaxPercentCacheForHitRatioVsCorpusSize_en_0.75_1
EmpLocMaxPercentCacheForHitRatioVsCorpusSize_en_0.75_2
EmpLocMaxPercentCacheForHitRatioVsCorpusSize_en_0.85_1
EmpLocMaxPercentCacheForHitRatioVsCorpusSize_en_0.85_2
EmpLocMaxPercentCacheForHitRatioVsCorpusSize_en_0.95_1
EmpLocMaxPercentCacheForHitRatioVsCorpusSize_en_0.95_2
```

EmpLocMaxPercentCacheForHitRatioVsCorpusSize_en_0.99_1
EmpLocMaxPercentCacheForHitRatioVsCorpusSize_en_0.99_2
EmpLocalMaxCacheForCorpusSizeVsHitRatio_en_11344756233_2
EmpLocalMaxCacheForCorpusSizeVsHitRatio_en_172051234833_2
EmpLocalMaxCacheForCorpusSizeVsHitRatio_en_31487751867_2
EmpLocalMaxCacheForCorpusSizeVsHitRatio_en_366397191_2
EmpLocalMaxCacheForCorpusSizeVsHitRatio_en_82718285958_2
EmpLocalMaxPercentCacheForCorpusSizeVsHitRatio_en_11344756233_2
EmpLocalMaxPercentCacheForCorpusSizeVsHitRatio_en_172051234833_2
EmpLocalMaxPercentCacheForCorpusSizeVsHitRatio_en_31487751867_2
EmpLocalMaxPercentCacheForCorpusSizeVsHitRatio_en_366397191_2
EmpLocalMaxPercentCacheForCorpusSizeVsHitRatio_en_82718285958_2
FirstDkRatioForCorpora_Emp_16_en_1
FirstDkRatioForCorpora_Emp_2_en_1
FirstDkRatioForCorpora_Emp_4_en_1
FirstDkRatioForCorpora_Emp_8_en_1
FirstDkRatioForCorpora_Pred_16_en_1
FirstDkRatioForCorpora_Pred_2_en_1
FirstDkRatioForCorpora_Pred_4_en_1
FirstDkRatioForCorpora_Pred_8_en_1
LocMaxCacheForHitRatioVsCorpusSize_en_0.75_1
LocMaxCacheForHitRatioVsCorpusSize_en_0.75_2
LocMaxCacheForHitRatioVsCorpusSize_en_0.85_1
LocMaxCacheForHitRatioVsCorpusSize_en_0.85_2
LocMaxCacheForHitRatioVsCorpusSize_en_0.95_1
LocMaxCacheForHitRatioVsCorpusSize_en_0.95_2
LocMaxCacheForHitRatioVsCorpusSize_en_0.99_1
LocMaxCacheForHitRatioVsCorpusSize_en_0.99_2
LocMaxPercentCacheForHitRatioVsCorpusSize_en_0.75_1
LocMaxPercentCacheForHitRatioVsCorpusSize_en_0.75_2
LocMaxPercentCacheForHitRatioVsCorpusSize_en_0.85_1
LocMaxPercentCacheForHitRatioVsCorpusSize_en_0.85_2
LocMaxPercentCacheForHitRatioVsCorpusSize_en_0.95_1
LocMaxPercentCacheForHitRatioVsCorpusSize_en_0.95_2
LocMaxPercentCacheForHitRatioVsCorpusSize_en_0.99_1
LocMaxPercentCacheForHitRatioVsCorpusSize_en_0.99_2
LocalMaxCacheForCorpusSizeVsHitRatio_en_11344756233_2
LocalMaxCacheForCorpusSizeVsHitRatio_en_172051234833_2
LocalMaxCacheForCorpusSizeVsHitRatio_en_31487751867_2
LocalMaxCacheForCorpusSizeVsHitRatio_en_366397191_2
LocalMaxCacheForCorpusSizeVsHitRatio_en_82718285958_2
LocalMaxEvolForFindBestCacheVsHitRatioPointForCorpusSize_en_40000000000.0_1_1
LocalMaxEvolForFindBestCacheVsHitRatioPointForCorpusSize_en_40000000000.0_1_2
LocalMaxFindBestCacheVsHitRatioPointForCorpusSize_en_40000000000.0_1_1
LocalMaxFindBestCacheVsHitRatioPointForCorpusSize_en_40000000000.0_1_2

LocalMaxPercentCacheForCorpusSizeVsHitRatio_en_11344756233_2
LocalMaxPercentCacheForCorpusSizeVsHitRatio_en_172051234833_2
LocalMaxPercentCacheForCorpusSizeVsHitRatio_en_31487751867_2
LocalMaxPercentCacheForCorpusSizeVsHitRatio_en_366397191_2
LocalMaxPercentCacheForCorpusSizeVsHitRatio_en_82718285958_2
GerFigFirstkDistinctNgramsRatio_en_1g.txt
GerLocMaxCacheForHitRatioVsCorpusSize_en_WithOutBloomF.txt
GerLocMaxPercentCacheForHitRatioVsCorpusSize_en_WithBloomF.txt
GerLocMaxPercentCacheForHitRatioVsCorpusSize_en_WithOutBloomF.txt
GerLocalMaxEvolForFindBestCacheVsHitRatioPointForCorpusSize_en_40000000000_1_1.txt
GerLocalMaxPercentCacheForCorpusSizeVsHitRatio_en_WithBloomF.txt

2 How results were obtained

In this section we explain how each Figure and results in the paper was generated.

2.1 Ratio of distinct 1-grams occurring less than k times, relative to the total number of distinct ones – Fig 1

By running the function *FirstDkRatioForCorpora* (in file `CacheDesignForLocalMax.py`) for English 1-grams and for the *corpora* size range from 3×10^7 to 3×10^{12} words, separately for k values less than 2, 4, 8 and 16, that is,

FirstDkRatioForCorpora('en 1, 2, CorpusSizeLimInf=3e7, CorpusSizeLimSup=3e12)

FirstDkRatioForCorpora('en 1, 4, CorpusSizeLimInf=3e7, CorpusSizeLimSup=3e12)

FirstDkRatioForCorpora('en 1, 8, CorpusSizeLimInf=3e7, CorpusSizeLimSup=3e12)

FirstDkRatioForCorpora('en 1, 16, CorpusSizeLimInf=3e7, CorpusSizeLimSup=3e12)

It produces the files

FirstDkRatioForCorpora_Pred_2_en_1
FirstDkRatioForCorpora_Emp_2_en_1
FirstDkRatioForCorpora_Pred_4_en_1
FirstDkRatioForCorpora_Emp_4_en_1
FirstDkRatioForCorpora_Pred_8_en_1
FirstDkRatioForCorpora_Emp_8_en_1
FirstDkRatioForCorpora_Pred_16_en_1
FirstDkRatioForCorpora_Emp_16_en_1 .

These last files are used to generate Figure 1 of the paper (`FirsKDistincNgramsRatio_en_1g.eps`), by running file

`GerFigFirstkDistinctNgramsRatio_en_1g.txt` in *gnuplot* prompt, that is, *gnuplot> load* `GerFigFirstkDistinctNgramsRatio_en_1g.txt` .

Furthermore, while running one of the cases, for example, *FirstDkRatioForCorpora('en 1, 4, CorpusSizeLimInf=3e7, CorpusSizeLimSup=3e12)*, it prints the relative errors for each test corpus:

Prediction for the ratio of number of first 3 distinct n-grams over D for Corpus 366397191 : 0.7907795685460486. Empirical: 0.7875648783126498. (Relative Error: 0.0040818100475560355

Prediction for the ratio of number of first 3 distinct n-grams over D for Corpus 11344756233 : 0.7940190436657513. Empirical: 0.7973389651597089. (Relative Error: 0.004163751727965116

Prediction for the ratio of number of first 3 distinct n-grams over D for Corpus 31487751867 : 0.7941224491885197. Empirical: 0.7939425255684908. (Relative Error: 0.00022662045958567558

Prediction for the ratio of number of first 3 distinct n-grams over D for Corpus 82718285958 : 0.7962679621229566. Empirical: 0.7949950803480521. (Relative Error: 0.001601119058934559

Prediction for the ratio of number of first 3 distinct n-grams over D for Corpus 172051234833 : 0.7928595737358247. Empirical: 0.7914959448340341. (Relative Error: 0.0017228501430624286

Avegare Relative Errors: 0.002359230287420763

2.2 Predicted and empirical global cache values, as a function of the *corpus* size, for different hit ratio values (0.75, 0.85, 0.95, 0.99), using no Bloom filters – Fig. 2

By running the function *EvalLocalMaxRealAndPercentCacheForHitRatioVsCorpusSizeWithOutBloomF* (in file `CacheDesignForLocalMax.py`) for English, using 1-grams to 6-grams and for the *corpora* size range from 4×10^8 to 4×10^{11} words with 50 steps, separately for hit-ratio values (0.75, 0.85, 0.95, 0.99), that is,

EvalLocalMaxRealAndPercentCacheForHitRatioVsCorpusSizeWithOutBloomF('en 6,0.75,4e8,4e11,50,verb=True)

EvalLocalMaxRealAndPercentCacheForHitRatioVsCorpusSizeWithOutBloomF('en 6,0.85,4e8,4e11,50,verb=True)

EvalLocalMaxRealAndPercentCacheForHitRatioVsCorpusSizeWithOutBloomF('en 6,0.95,4e8,4e11,50,verb=True)

EvalLocalMaxRealAndPercentCacheForHitRatioVsCorpusSizeWithOutBloomF('en 6,0.99,4e8,4e11,50,verb=True) ,

it produces the files

`LocMaxCacheForHitRatioVsCorpusSize_en_0.75_1`
`EmpLocMaxCacheForHitRatioVsCorpusSize_en_0.75_1`
`LocMaxCacheForHitRatioVsCorpusSize_en_0.85_1`

`EmpLocMaxCacheForHitRatioVsCorpusSize_en_0.85_1`
`LocMaxCacheForHitRatioVsCorpusSize_en_0.95_1`
`EmpLocMaxCacheForHitRatioVsCorpusSize_en_0.95_1`
`LocMaxCacheForHitRatioVsCorpusSize_en_0.99_1`
`EmpLocMaxCacheForHitRatioVsCorpusSize_en_0.99_1`.

These files are used to generate Figure 2 of the paper (`LocMaxCacheForHitRatioVsCorpusSize_en_WithOutBloomF.eps`), after running in the *gnuplot* prompt,
gnuplot> *load* `GerLocMaxCacheForHitRatioVsCorpusSize_en_WithOutBloomF.txt`.

Besides, while running one of the cases, for example,
EvalLocalMaxRealAndPercentCacheForHitRatioVsCorpusSizeWithOutBloomF('en 6,0.95,4e8,4e11,50,verb=True), it also prints the relative errors obtained for each test corpus:

Relative Error of LocalMax cache size for HitRatio 0.95 and corpus size 366397191 : 0.00990597096447563
Relative Error of LocalMax cache size for HitRatio 0.95 and corpus size 11344756233 : 0.011629639685973005
Relative Error of LocalMax cache size for HitRatio 0.95 and corpus size 31487751867 : 0.005166363015926882
Relative Error of LocalMax cache size for HitRatio 0.95 and corpus size 82718285958 : 0.01793819088845537
Relative Error of LocalMax cache size for HitRatio 0.95 and corpus size 172051234833 : 0.003724417534384178
Avg Error 0.009672916417843012

2.3 Finding the Cache Elasticity Minimum Point (CEMP), using no Bloom filters – Fig. 3

In order to predict the CEMP for a 40 billion word English *corpus*, as well as the evolution of the hit ratio as a function of the number of entries of the global cache (for 1-grams to 6-grams), when using no Bloom filters, the function *LocalMaxFindBestCacheVsHitRatioPointForCorpusSizeWithOutBloomF* (in file `CacheDesignForLocalMax.py`) must run, that is,
LocalMaxFindBestCacheVsHitRatioPointForCorpusSizeWithOutBloomF('en', 6,4e10,0.65,0.0025,CacheImportance=1,verb=True), where 0.65 is the starting point for hit ratio, and 0.0025 is the value for $\Delta HitRatio$. It produces the file
`LocalMaxEvolForFindBestCacheVsHitRatioPointForCorpusSize_en_40000000000.0_1_1`
which is used when running, in gnuplot,
gnuplot> *load*
`GerLocalMaxEvolForFindBestCacheVsHitRatioPointForCorpusSize_en_40000000000_1_1.txt`
to produce Figure 3 of the paper
(`LocalMaxEvolForFindBestCacheVsHitRatioPointForCorpusSize_en_4e10_1_1.eps`).

While the

LocalMaxFindBestCacheVsHitRatioPointForCorpusSizeWithOutBloomF('en', 6,4e10,0.65,0.0025,CacheImportance=1,verb=True)

is running, it prints the prediction of the CEMP for the 40 billion word *corpus*:

Most efficient HitRatio Vs Cache size for LocalMax application, for 40000000000.0 corpus size : 0.9274999999999941 (Hit Ratio); 20065236111.98228 (Cache Size) corresponding to 0.6118918126508907 of cache size for 100% Hit ratio.

2.4 Ratio of global cache size with respect to full cache size for different hit ratio values (0.75, 0.85, 0.95, 0.99), using no Bloom filters – Fig. 4

By running the function *EvalLocalMaxRealAndPercentCacheForHitRatioVsCorpusSizeWithOutBloomF* (in file `CacheDesignForLocalMax.py`) for English, using 1-grams to 6-grams and for the *corpora* size range from 4×10^8 to 4×10^{11} words with 50 steps, separately for hit-ratio values (0.75, 0.85, 0.95, 0.99), as it was done in Subsection (2.2), that is,

EvalLocalMaxRealAndPercentCacheForHitRatioVsCorpusSizeWithOutBloomF('en 6,0.75,4e8,4e11,50,verb=True)

EvalLocalMaxRealAndPercentCacheForHitRatioVsCorpusSizeWithOutBloomF('en 6,0.85,4e8,4e11,50,verb=True)

EvalLocalMaxRealAndPercentCacheForHitRatioVsCorpusSizeWithOutBloomF('en 6,0.95,4e8,4e11,50,verb=True)

EvalLocalMaxRealAndPercentCacheForHitRatioVsCorpusSizeWithOutBloomF('en 6,0.99,4e8,4e11,50,verb=True) ,

it also produces the files

`LocMaxPercentCacheForHitRatioVsCorpusSize_en_0.75_1`
`EmpLocMaxPercentCacheForHitRatioVsCorpusSize_en_0.75_1`
`LocMaxPercentCacheForHitRatioVsCorpusSize_en_0.85_1`
`EmpLocMaxPercentCacheForHitRatioVsCorpusSize_en_0.85_1`
`LocMaxPercentCacheForHitRatioVsCorpusSize_en_0.95_1`
`EmpLocMaxPercentCacheForHitRatioVsCorpusSize_en_0.95_1`
`LocMaxPercentCacheForHitRatioVsCorpusSize_en_0.99_1`
`EmpLocMaxPercentCacheForHitRatioVsCorpusSize_en_0.99_1` .

These files are used to generate Figure 4 of the paper

(`LocMaxPercentCacheForHitRatioVsCorpusSize_en_WithOutBloomF.eps`),
after running in the *gnuplot* prompt,

gnuplot> *load* `GerLocMaxPercentCacheForHitRatioVsCorpusSize_en_WithOutBloomF.txt`
.

2.5 Ratio of global cache size with respect to full cache size, as a function of *corpus* size, for different hit ratio values (0.75, 0.85, 0.95, 0.99), using Bloom filters – Fig.5

By running the function *EvalLocalMaxRealAndPercentCacheForHitRatioVsCorpusSizeWithBloomF* (in file `CacheDesignForLocalMax.py`) for English, using 1-grams to 6-grams and for the *corpora* size range from 4×10^8 to 4×10^{11} words with 50 steps, separately for hit-ratio values (0.75, 0.85, 0.95, 0.99), that is,

EvalLocalMaxRealAndPercentCacheForHitRatioVsCorpusSizeWithBloomF('en 6,0.75,4e8,4e11,50,verb=True)

EvalLocalMaxRealAndPercentCacheForHitRatioVsCorpusSizeWithBloomF('en 6,0.85,4e8,4e11,50,verb=True)

EvalLocalMaxRealAndPercentCacheForHitRatioVsCorpusSizeWithBloomF('en 6,0.95,4e8,4e11,50,verb=True)

EvalLocalMaxRealAndPercentCacheForHitRatioVsCorpusSizeWithBloomF('en 6,0.99,4e8,4e11,50,verb=True) .

It produces the files

`LocMaxPercentCacheForHitRatioVsCorpusSize_en_0.75_2`
`EmpLocMaxPercentCacheForHitRatioVsCorpusSize_en_0.75_2`
`LocMaxPercentCacheForHitRatioVsCorpusSize_en_0.85_2`
`EmpLocMaxPercentCacheForHitRatioVsCorpusSize_en_0.85_2`
`LocMaxPercentCacheForHitRatioVsCorpusSize_en_0.95_2`
`EmpLocMaxPercentCacheForHitRatioVsCorpusSize_en_0.95_2`
`LocMaxPercentCacheForHitRatioVsCorpusSize_en_0.99_2`
`EmpLocMaxPercentCacheForHitRatioVsCorpusSize_en_0.99_2` .

These files are used to generate Figure 5 of the paper (`LocMaxPercentCacheForHitRatioVsCorpusSize_en_WithBloomF.eps`), after running in the *gnuplot* prompt,

gnuplot> *load* `GerLocMaxPercentCacheForHitRatioVsCorpusSize_en_WithBloomF.txt`.

Furthermore, while running one of the cases, for example, *EvalLocalMaxRealAndPercentCacheForHitRatioVsCorpusSizeWithBloomF('en 6,0.95,4e8,4e11,50,verb=True)*, it also prints the relative errors obtained for each test corpus:

Relative Error of LocalMax cache size for HitRatio 0.95 and corpus size 366397191 : 0.004636328842249107

Relative Error of LocalMax cache size for HitRatio 0.95 and corpus size 11344756233 : 0.041352682483483005

Relative Error of LocalMax cache size for HitRatio 0.95 and corpus size 31487751867 : 0.000591719361396837

Relative Error of LocalMax cache size for HitRatio 0.95 and corpus size 82718285958 : 0.009329605207396802

Relative Error of LocalMax cache size for HitRatio 0.95 and corpus size 172051234833 : 0.005259150707476063

Avg Error 0.012233897320400363

2.6 Ratio of global cache size with respect to full cache size, as a function of hit ratio, using Bloom filters – Fig. 6

By running the function *EvalLocalMaxRealAndPercentCacheForCorpusSizeVsHitRatioWithBloomF* (in file `CacheDesignForLocalMax.py`) for English, using 1-grams to 6-grams, for 50 hit ratio step values starting from 0.75, for different *corpus* sizes, that is,

EvalLocalMaxRealAndPercentCacheForCorpusSizeVsHitRatioWithBloomF('en', 6,366397191,0.75,50,verb=True)

EvalLocalMaxRealAndPercentCacheForCorpusSizeVsHitRatioWithBloomF('en', 6,11344756233,0.75,50,verb=True)

EvalLocalMaxRealAndPercentCacheForCorpusSizeVsHitRatioWithBloomF('en', 6,31487751867,0.75,50,verb=True)

EvalLocalMaxRealAndPercentCacheForCorpusSizeVsHitRatioWithBloomF('en', 6,82718285958,0.75,50,verb=True)

EvalLocalMaxRealAndPercentCacheForCorpusSizeVsHitRatioWithBloomF('en', 6,172051234833,0.75,50,verb=True),

it produces the files

`LocalMaxCacheForCorpusSizeVsHitRatio_en_366397191_2`
`EmpLocalMaxPercentCacheForCorpusSizeVsHitRatio_en_366397191_2`
`LocalMaxCacheForCorpusSizeVsHitRatio_en_11344756233_2`
`EmpLocalMaxPercentCacheForCorpusSizeVsHitRatio_en_11344756233_2`
`LocalMaxCacheForCorpusSizeVsHitRatio_en_31487751867_2`
`EmpLocalMaxPercentCacheForCorpusSizeVsHitRatio_en_31487751867_2`
`LocalMaxCacheForCorpusSizeVsHitRatio_en_82718285958_2`
`EmpLocalMaxPercentCacheForCorpusSizeVsHitRatio_en_82718285958_2`
`LocalMaxCacheForCorpusSizeVsHitRatio_en_172051234833_2`
`EmpLocalMaxPercentCacheForCorpusSizeVsHitRatio_en_172051234833_2`.

These files are used to generate Figure 6 of the paper (`LocalMaxPercentCacheForCorpusSizeVsHitRatio_en_WithBloomF.eps`), after running in the *gnuplot* prompt,

gnuplot> load

`GerLocalMaxPercentCacheForCorpusSizeVsHitRatio_en_WithBloomF.txt`.

Furthermore, while running one of the cases, for example, *EvalLocalMaxRealAndPercentCacheForCorpusSizeVsHitRatioWithBloomF('en', 6,82718285958,0.75,50,verb=True)*, it prints the relative error for the test *corpus*, for each hit ratio value, for example:

...
...

Relative Error of cache size for HitRatio 0.9950397263139075 and corpus size 82718285958 : 0.01323376549391151

Relative Error of cache size for HitRatio 0.9954210902778164 and corpus size 82718285958 : 0.012858306064061674

Mean Error: 0.013493339316773937 ; Mean Square Root Error: 0.016446999441402883

2.7 Finding the Cache Elasticity Minimum Point (CEMP), using Bloom filters

In order to predict the CEMP for a 40 billion word English *corpus*, as well as the evolution of the hit ratio as a function of the number of entries of the global cache (for 1-grams to 6-grams), when using Bloom filters, the function

LocalMaxFindBestCacheVsHitRatioPointForCorpusSizeWithBloomF (in file `CacheDesignForLocalMax.py`) must run, that is,

LocalMaxFindBestCacheVsHitRatioPointForCorpusSizeWithBloomF('en', 6, 4e10, 0.75, 0.0025, CacheImportance=1, verb=True), where 0.75 is starting point for hit ratio, and 0.0025 is the value for $\Delta HitRatio$. It produces the file

`LocalMaxEvolForFindBestCacheVsHitRatioPointForCorpusSize_en_40000000000.0_1_2` containing the mentioned evolution. When the function is running, it prints the prediction of the CEMP for the 40 billion word *corpus* when using Bloom filters:

Most efficient HitRatio Vs Cache size for LocalMax application, for 40000000000.0 corpus size : 0.8074999999999988 (Hit Ratio); 2886573848.7816358 (Cache Size) corresponding to 0.08802642016392201 of cache size for 100% Hit ratio.

2.8 The prediction model

The software implementing the statistical prediction model, which is one of the pillars supporting the approach proposed in the paper, can be accessed at https://github.com/OurName1234/ngrams .