CS 224N: Language Models (PA1)

Sandeep Sripada(ssandeep) & **Venu Gopal Kasturi(venuk)**

1 Introduction

In Programming Assignment-1, the task was to implement Language models (LMs) using Europarl corpus. The LMs help us to model the probabilities of a sentence occurring in a language using the conditional probabilities of individual words and their context. We used n-gram models to model the language and the various n-grams tried include unigrams, bigrams and trigrams. The performance increased as more context (history) was taken into account. However, models beyond trigrams were not taken up because of the computational blowup and marginal improvement.

We used various smoothing techniques to get effective and accurate models. We started by looking at techniques like Maximum likelihood estimate (MLE), Add-1 and Add-δ, which did not prove to be effective. So improving upon those, we implemented Simple Good-Turing (SGT), Absolute Discounting (AD) methods for our n-grams. We also implemented improvements upon the stand alone models by using: (1) backoff methods like Katz, Kneser-Ney, Witten-Bell to 'back-off' to lower order models when the conditional distributions did not exist, and (2) interpolation methods where the weights of models was determined by using Expectation-Maximization algorithm on held-out data. We also experimented with various combinations of well performing models in linear interpolation to generate weights accordingly and testing the perplexity of the hybrid model.

Our best model based on test perplexity was a linear interpolation between Simple Good-Turing unigram model, Katz bigram and Kneser-Ney trigram models. When training on the 40,000 sentence corpus, the model has a test perplexity of **166.11**. Our best model based on the Enron mail recovery task was a combination of the above mentioned models with Enron WER at **0.11** and Enron percent correct at **75**. When training on all of the available data ($\sim$ 1 million sentences), the best model was the interpolation of Simple Good-Turing unigram model, Knerser-Ney Bigram and Trigram models and had a test perplexity of **99.28**, Enron WER **0.065** and % correct **85.34**.

2 Implementation details and Setup

The most naïve Language model is based on the principle of Maximum Likelihood Estimation which maximizes the probability of the data given the model. But the problem with this kind of model is that it assigns zero to the probability of seeing an unseen event. This is also called as the data sparsity problem and there are various techniques to build Language Models that try to circumvent the issue by smoothing the probability distributions.

The various smoothing methods[1] we tried include:

1. Maximum Likelihood Estimation
2. Add-One Smoothing
3. Add-Delta Smoothing
4. Absolute Discounting
5. Simple Good-Turing Discounting: This is based on the algorithm proposed by Good (credited Turing) in 1953 which estimates the probabilities using the frequency of frequency measure.
6. Katz Back-off Model (1987)
7. Linear Interpolation weight assignment done using Expectation Maximization
8. Interpolated Kneser-Ney smoothing model (1995)
9. Absolute discounting involving interpolation of lower and higher order models
10. Witten-Bell discounting (1991)

Simple Good-Turing method gave a better estimation for unknown events and hence performed well when compared to other methods such as Add-1, Add-δ, MLE etc. We also implemented a linear regression based count estimation for lower order frequency counts i.e. we used a log-log plot of frequency vs 'frequency of frequency' to accurately estimate counts of frequencies below a certain threshold (5 in our case).

Following the experimental results mentioned in [Chen1998] and [Goodman2000], we also implemented back-off based models like Kneser-Ney, Absolute

[1] Overview of the methods can be found here: `http://nlp.stanford.edu/~wcmac/papers/20050421-smoothing-tutorial.pdf`

Table 1: Best model results for Test perplexity (40k and 1 million training sentences), Enron WER and % correct

Model	Data	Perplexity	WER	% cor
Katz-Bi,KN-Tri	40k	166.11	0.11	75%
KN-Bi,KN-Tri	1 mill	99.28	0.065	85.34%
Katz-Bi,KN-Tri	1 mill	99.43	0.063	86.21%

Discounting, Witten-Bell etc. which performed better than the simple stand-alone models.

The data used for training the language models for both tasks was based on Europarl corpus. The size of the data set is around 1.15 million sentences which was split into training, test and validate datasets. The original Europarl lowercased data was cleaned and sentences with 39 or more tokens were filtered out resulting in 1,140,475 sentences. The other data used was from Enron which had the first 5000 lines after processing.

Naming: The files are named using the prefix SGT for all Simple Good-Turing based models, MLE for Maximum Likelihood Estimation models, AD for simple absolute discounting model. All the other methods have the their name included in the filename (eg: WittenBellSmoothingModel (WB), SimpleTrigramInterpolation (LinInp), KneserNeyModel (KN), KatzBackoffModel (Katz), AbsDiscModel (AD)). These methods are from here on referred to by the signature mentioned in the parenthesis.

3 Language Models

Proofs showing that the model satisfies a probability distribution i.e. $\sum_w P(w|h) = 1$ are attached at the end.

4 Method details

Although, simple Good-Turing performed better than Add-1, Add-δ, MLE methods, it was not definitely the best smoothing method. As show in the following section, the back-off and interpolated models performed much better than stand-along n-gram models.

Kneser-Ney smoothing: This method is an extension of absolute discounting with a clever way of constructing the lower-order (backoff) model. The lower-order model is significant only when count is small or zero in the higher-order model, and so should be optimized for that purpose.

$p_{KN}(w_i|w_{i-n+1}^{i-1}) = \frac{max(c(w_{i-n+1}^{i})-\delta,0)}{\sum_{w_i} c(w_{i-n+1}^{i})} + \frac{\delta}{\sum_{w_i c(w_{i-n+1}^{i})}} N_{1+}(w_{i-n+1}^{i-1}*) p_{KN}(w_i|w_{i-n+2}^{i-1})$

with

$p_{KN}(w_i) = \frac{N_{1+}(*w_i)}{N_{1+}(**)}$

and

$N_{1+}(*w_i) = |w_{i-1} : c(w_{i-1}w_i) > 0|$

Absolute Discounting: This method involves interpolation of lower and higher order models.

$p_{abs}(w_i|w_{i-n+1}^{i-1}) = \frac{max(c(w_{i-n+1}^{i})-\delta,0)}{\sum_{w_i} c(w_{i-n+1}^{i})} + (1 - \lambda_{w_{i-n+1}^{i-1}}) p_{abs}(w_i|w_{i-n+1}^{i-1})$

and

$(1 - \lambda_{w_{i-n+1}^{i-1}}) = \frac{\delta}{\sum w_i c(w_{i-n+1}^{i})} N_{1+}(w_{i-n+1}^{i-1}*)$

Witten-Bell smoothing: A similar backoff model with auto interpolation of a backed-off model.

$p_{WB}(w^i|w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} p_{ML}(w_{i-n+1}^{i-1}) + (1 - \lambda_{w_{i-n+1}^{i-1}}) p_{WB}(w_{i-n+2}^{i-1})$ where $\lambda_{w_{i-n+1}^{i-1}}$ can be interpreted as the probability of using the higher order model and is calculated appropriately to make it a probability distribution.

$1 - \lambda_{w_{i-n+1}^{i-1}} = \frac{N_{1+}(w_{i-n+1}^{i-1}*)}{N_{1+}(w_{i-n+1}^{i-1}*) + \sum_{w_i} c(w_{i-n+1}^{i})}$

Katz model: This method is based on the paper [Katz87].

Linear Interpolation: This method helps in setting weights by running on a completely new dataset (held-out data). It has 2 steps (E-step and M-step) which are run until a termination criteria is met. Steps: (1) calculate the likelihood of various completions with our fixed θ', and (2) maximize the expected log-likelihood of the complete data. The termination conditions we used are: (1) The euclidean distance between old and new θ's is less than e^{-4}, and/or (2) the number of iterations if less than 50.

Handling Unknowns: We have introduced an UNKNOWN word which encompasses all the unseen unigrams in the test data. Hence our Vocabulary is the number of different unigrams in the training set + UNKNOWN + STOP + START. For the Unigram model, we are returning the probability of seeing an UNKNOWN word for any new word in the test data.

For the Bigram and Trigram models, we are transforming the unseen word in the test data into an UNKNOWN word and then calculating the conditional probability of the N-gram, by distributing the mass of the total mass of the unseen events to these each individual new n-grams.

For Simple Good Turing smoothing: The probability mass of the total unseen events is divided amongst all unseen events equally. We have estimated the number of unseen events to be : $O(V^N)$ - number of unique n-grams seen in the training set.

Table 2: Best model results (perplexity, WER and % correct) for Linear Interpolation Models (40k) (Unigram model in all cases: SGT)

Interpolation	Perplexity	WER	% correct
WB-Bi, KN-Tri	166.48	0.1257	72.42
WB-Bi, Katz-Tri	182.26	0.133	67.24
AD-Bi, KN-Tri	166.11	0.1226	73.27
KN-Bi, AD-Tri	167.92	0.122	75
WB-Bi, AD-Tri	170.55	0.1357	70.68
Katz-Bi, KN-Tri	166.11	0.11	75

Table 3: Best model results (perplexity, WER and % correct) for Linear Interpolation Models (1 million) (Unigram model in all cases: SGT)

Interpolation	Perplexity	WER	% correct
KN-Bi, KN-Tri	99.28	0.065	85.34
Katz-Bi, KN-Tri	99.43	0.063	86.21
AD-Bi, KN-Tri	99.53	0.063	86.21

5 Performance details

5.1 40,000 sentence training set

The test set perplexity for some of the models when trained on the 40k dataset is as shown in Figure 1. The best models were obtained using a linear interpolation of various back-off models. The weights for linear interpolation are maximized using the corresponding models on held-out data. The best model is the linear interpolation of Simple Good-Turing unigram model, Katz bigram and Kneser-Ney trigram models (also the combination of Simple Good-Turing unigram model, Absolute Discounting bigram model and Kneser-Ney trigram gave the same result). This model obtained a test perplexity of **166.11**. Interpolating models which use the maximum possible context (upto trigrams) is almost always better than interpolating models that do not fully utilize the entire context (unigram, bigram). Also, we tried with various combinations for linear interpolation of back-off models as shown in Table 2 and the performance of most of the combinations is high and around a similar range (avg: 169.9).

Some of the models not included in the figure are: SGT base models(uni, bi, tri), MLE base models (uni, bi, tri), KatzTrigram as they had high test perplexities. The SGTTrigram model did not perform well because the model assigned a very low probability to unknown events and hence while testing on a new dataset the probability of a sentence was very low. The unknown probability mass was distributed evenly among very large number of unknown possibilities and hence the reason for assigning low probabilities.

5.2 1 million sentence training set

The test set perplexity for some of the models when trained on the 1 million dataset is as shown in Figure 2. The best models were obtained using a linear interpolation of various back-off models. The weights for linear interpolation are maximized using the corresponding models on held-out data. The best model is the linear interpolation of Simple Good-Turing unigram model, Witten-Bell bigram model and Kneser-Ney trigram model. This model obtained a test perplexity of **99.28**. Interpolating models which use the maximum possible context (upto trigrams) is almost always better than interpolating models that do not fully utilize the entire context (unigram, bigram). Also, we tried with various combinations for linear interpolation of back-off models as shown in Table 3 and the performance of most of the combinations is high and around a similar range (avg: 99.41).

Also, the difference between the perplexities of various combinations was very small and this shows that as the training set size increases, the use of smoothed models within interpolation becomes less important. This is expected because with a larger datasets we can trust the counts more. For the same reasons as mentioned in the above section, the base models were not included as they had high perplexity values.

When using the Interpolation models for 1 Million data using combinations of Kneser-Ney, Absolute Discounting and Katz Bigram models, we have noticed that all of them are performing almost similarly. And also we have noticed that the increase in accuracy of the models when compared to the performance of Kneser-Ney Trigram model on the same training set is very less indicating the power of using just a backing off model (which inherently has an interpolation component).

5.3 Learning curves

We also performed a detailed analysis on how the models preform when the training set is gradually incremented from 40k to 1 million. Figure shows the behavior of some of the models. As data increases, the test perplexity decreases as more information is present (thereby trust the counts) and lower number of unseen combinations (thereby increasing the probability of unknown events). The learning curves are as shown in Figure **??**.

5.4 Unigram Vs Bigram Vs Trigram

The performance of the base models improved as more context is taken into account with the exception of trigrams. SGTTrigram almost perfectly does well on train-

ing data (overfitting) but fails miserably when it comes to the test data. This is because one million sentences is not enough to obtain accurate counts for each conditional probability distribution. All the other back-off models performed well as context increased except for KatzTrigram which performed extremely badly when compared to the others. The plots are shown in Figure 3. The models had very high perplexities and those models which have touched the 1500 mark have much higher values (around 17000) and have been truncated to make the plot readable.

Almost similar behavior is exhibited with the one million dataset but with lesser differences between the best and the worst models in case of trigrams. As more information is available better probability values are assigned and hence these models performed well. The trigram models performed well when compared to the 40k dataset as they had more information available. The results are as shown in Figure 4.

5.5 Enron data

The results are as shown in Figure 5 and Figure 6 for the 40k dataset. The word error rate is fairly consistent across the base models and is better in case of interpolation models. The best model has a WER of about 0.11 and 75% Enron % correct.

The results are as shown in Figure 7 and Figure 8 for the 1 million dataset. The word error rate is fairly consistent across the base models and is better in case of interpolation models. The best model has a WER of about 0.063 and 86.21% Enron % correct.

As expected the interpolation models were better at predicting the gaps in the mails and hence performed well. The WER was worst in models where context was not taken (again as expected) as filling a slot would just be done by the word with highest probability irrespective of the context.

5.6 Output Sentence Analysis

The sentences generated using the base unigram models made no sense at all as all they did was to select a word that had a high probability without using any context information. The sentences generated using the base bigram models was similar to the previous case. The sentences generated using the base trigram models however made some sense. Though the entire sentence was not perfect, there were section within the sentence that were grammatically correct.

Eg: A base trigram generated sentence: [(S), (S), genetic, intervention, prices, have, an, important, contribution, to, peace, (/S)]

The use of back-off and interpolation accentuated the above models performances. The back-off, interpolation based trigrams generated text that made most sense among all the other models. The performance of the models on Enron mail recovery was not as expected partly because of the different domains of the training and test data. However, for normal usage words, the mails recovered had some good completions.

Eg: A trigram model (AD) generated sentence: [(S), (S), it, is, high, time, that, we, are, taking, precedence, over, the, world, 's, committee, on, budgetary, control, (/S)]

The models perform bad once an unconventional word like a year, number appears as the models would then take the context into account and drift.

Eg: A bigram generated sentence: [(S), it, is, in, the, scandalous, between, lovers, of, both, inhumane, and, debate, on, the, fifth, paragraph, 13, day, of, 1953, in, (/S)]

For 1 million dataset, the sentences generated were of good quality.

Eg: AD-Trigram [(S), (S), a, source, of, agricultural, policy, is, still, needed, (/S)]
KN-Trigram [(S), (S), finally, we, need, to, reduce, the, differences, of, opinion, on, the, way, very, grateful, if, they, offer, a, half], [(S), (S), genetic, intervention, prices, have, an, important, contribution, to, peace, (/S)], [(S), (S), in, the, matter, of, real, human, lives, and, works, agency, is, all, about, (/S)].

6 Contributions

Venu Gopal Katuri: Worked on building the Unigram models with Simple Good Turing smoothing. Implemented the Simple Absolute Discounting for Unigrams, Katz Backoff model, Absolute Discounting, Kneser Nay Smoothing. Also worked on writing the proofs for Check Models for all the models implemented and the report.

Sandeep Sripada: Worked on building the Bigram Models with Simple Absolute Discounting and the Bi-,Tri- models with Simple Good Turing smoothing and MLE. Worked on building the Linear Interpolation models with Expectation Maximization and then implemented the Witten Bell smoothing for Bi and Tri grams. I have worked on writing the report and proofs

References

[Chen1998] An empirical study of smoothing techniques for language modeling.
Stanley Chen and Joshua Goodman.
Technical report TR-10-98, Harvard University, August 1998.

[Goodman2000] Putting It All Together: Language Model Combination.
Joshua T. Goodman.

In Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing. 2000.

[J&M] Speech and Language Processing.
Daniel Jurafsky and James h. Martin.
Second ed. Prentice Hall. (ISBN-10: 0131873210)

[Katz87] Estimation of probabilities from sparse data for the language model component of a speech recogniser.
S. M. Katz.
IEEE Transactions on Acoustics, Speech, and Signal Processing, 35(3), 400401

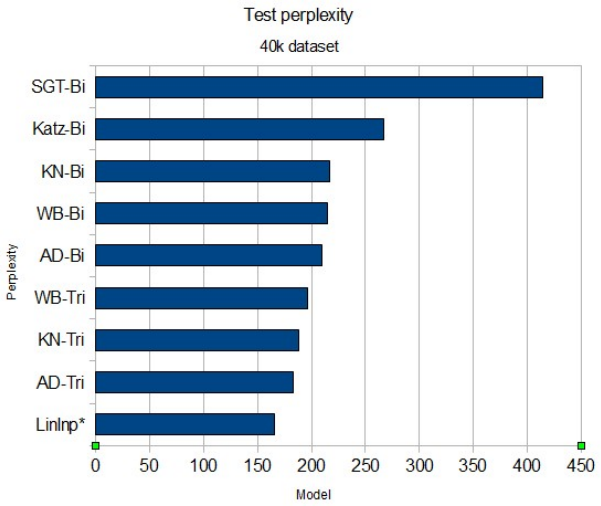

Figure 1: Best model test perplexity results for 40k dataset

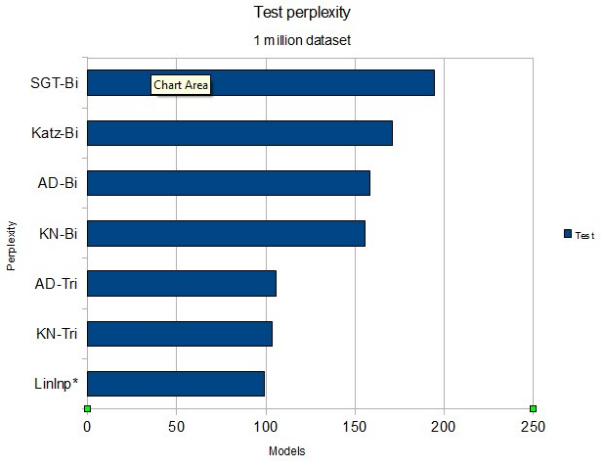

Figure 2: Best model test perplexity results for 1 million dataset

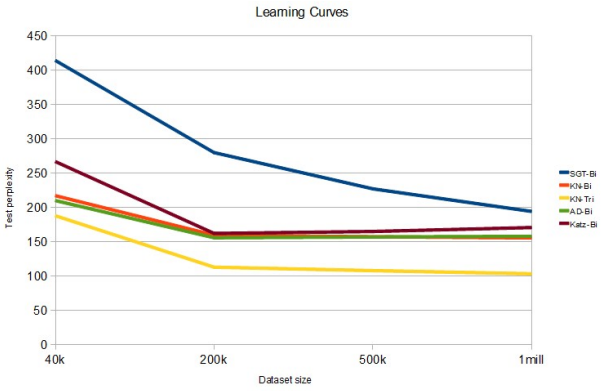

Figure 3: Learning curve for some models on the dataset

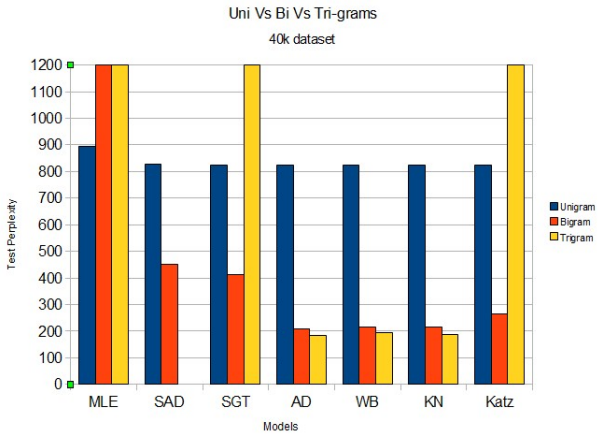

Figure 4: Comparison of n-gram perplexities as n varies (40k)

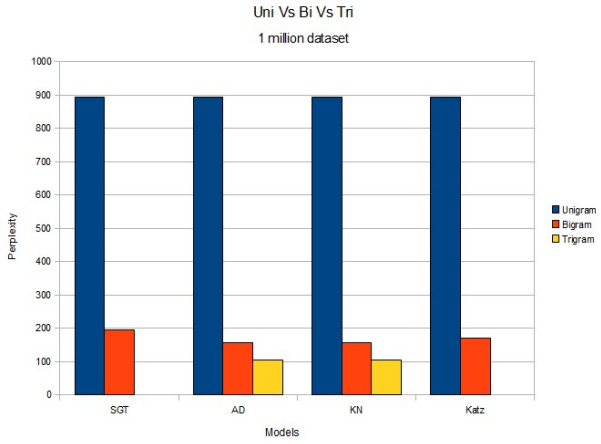

Figure 5: Comparison of n-gram perplexities as n varies (1 million)

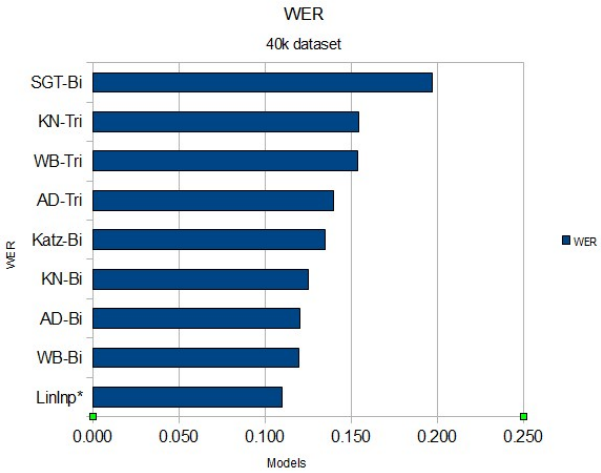

Figure 6: Enron WER for various models on 40k dataset

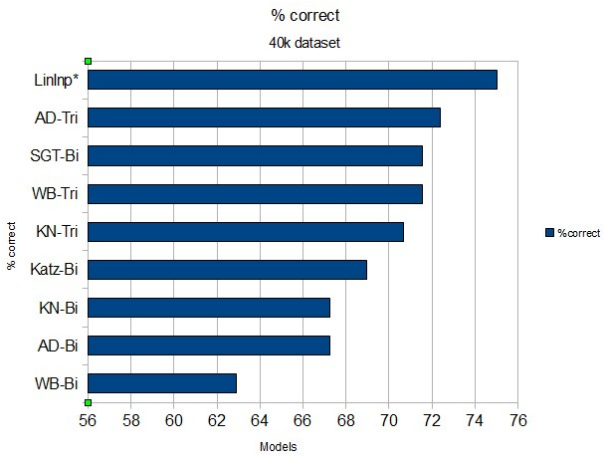

Figure 7: Enron % correct for various models on 40k dataset

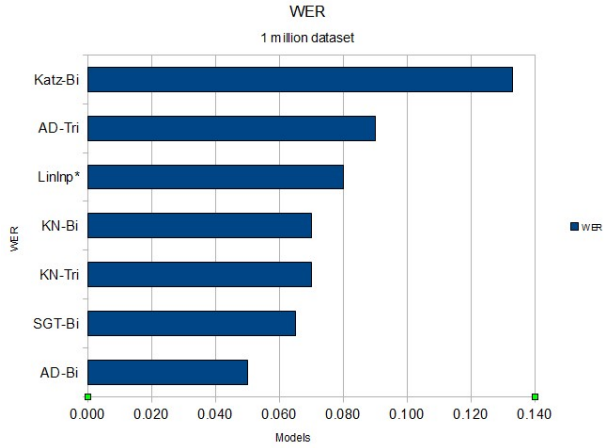

Figure 8: Enron WER for various models on 40k dataset

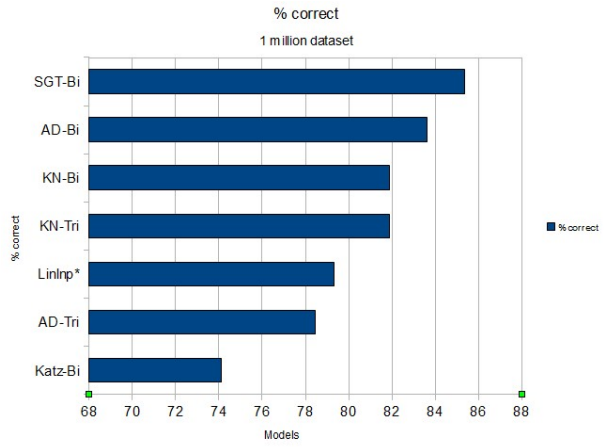

Figure 9: Enron % correct for various models on 40k dataset