1.

(a)

Unigram: 2 states, 28 arcs; **Bigram**: 30 states, 578 arcs; **Trigram**: 531 states, 4996 arcs

(b)

Unigram: 2^-39006.2; **Bigram**: 2^-32407.6; **Trigram**: 2^-27194.2

(c)

Unigram:

No smoothing. Both TRAIN and TEST are used as training data to collect letter frequency information.

Bigram:

Modified Kneser-Ney smoothing, which was briefly mentioned in class and is described in [Chen and Goodman 1996] and [James 2000]. Specifically, given that the frequency of bigram xy is c(xy), and that

$$\mathrm{c}(\mathrm{x}\cdot) = \sum_{\mathrm{y}'} c(xy')$$

$$\mathrm{N_c}(x\cdot) = |\{y' : c(xy') = c\}|$$
$$\mathrm{N_{1+}}(\cdot y) = |\{x' : c(x'y) > 0\}|$$
$$\mathrm{N_{1+}}(\cdot\cdot) = |\{x', y' : c(x'y') > 0\}|$$

the probability assigned to each bigram xy is:

$$\mathrm{P}_{\mathrm{KN}}^{(2)}(y|x) = \alpha^{(2)}(y|x) + \gamma^{(1)}(x) \cdot P_{KN}^{(1)}(y)$$

where

$$\alpha^{(2)}(y|x) = \frac{\mathrm{c(xy)} - D\big(c(xy)\big)}{c(x\cdot)}$$
$$\gamma^{(1)}(x) = \frac{\sum_{\mathrm{c}} D(c) N_c(x\cdot)}{c(x\cdot)}$$
$$\mathrm{P}_{\mathrm{KN}}^{(1)}(y) = \frac{N_{1+}(\cdot y)}{N_{1+}(\cdot\cdot)}$$

and

$$\mathrm{D} = \begin{cases} 0 \; if \; c = 0 \\ D_1 \; if \; c = 1 \\ D_2 \; if \; c = 2 \\ D_{3+} \; if \; c \geq 3 \end{cases}$$

The default values of D are set to:

$$\mathrm{Y} = \frac{\mathrm{n}_1}{n_1 + 2n_2}$$
$$\mathrm{D}_1 = 1 - 2Y\frac{n_2}{n_1}$$
$$\mathrm{D}_2 = 2 - 3Y\frac{n_3}{n_2}$$
$$\mathrm{D}_{3+} = 3 - 4Y\frac{n_4}{n_3}$$

where $\mathrm{n_j}$ is the number of bigrams that appear exactly j times. While after tuning on the HELDOUT data, $\mathrm{D}_1, \mathrm{D}_2$ and D_{3+} are finally set to **0.290**, **0.048** and **0.830** respectively. After the parameters are set, TRAIN and HELDOUT are combined as training data to retrain the model.

Trigram:

Modified Kneser-Ney smoothing is also used here. Specifically, similar to the case in bigram, the probability assigned to each trigram xyz is:

$$\mathrm{P}_{\mathrm{KN}}^{(3)}(z|xy) = \alpha^{(3)}(z|xy) + \gamma^{(2)}(xy) \cdot P_{KN}^{(2)}(z|y)$$

where

$$\mathrm{P}_{\mathrm{KN}}^{(2)}(z|y) = \alpha^{(2)}(z|y) + \gamma^{(1)}(y) \cdot P_{KN}^{(1)}(z)$$

The definition of $P_{KN}^{(1)}(z)$ is the same as before. The interpolation coefficients are defined as

$$\alpha^{(3)}(z|xy) = \frac{\mathrm{c(xyz)} - D^{(3)}\big(c(xyz)\big)}{c(xy\cdot)}$$
$$\gamma^{(2)}(x) = \frac{\sum_{\mathrm{c}} D^{(3)}(c) N_c(xy\cdot)}{c(xy\cdot)}$$
$$\alpha^{(2)}(z|y) = \frac{\mathrm{N_{1+}}(\cdot yz) - D^{(2)}\big(\mathrm{N_{1+}}(\cdot yz)\big)}{N_{1+}(\cdot\, y\,\cdot)} \; (if \; y \neq \text{^})$$
$$\alpha^{(2)}(x|\text{^}) = \frac{\mathrm{c(^x)} - D^{(2)}\big(c(\text{^x})\big)}{c(\text{^}\,\cdot)}$$
$$\gamma^{(1)}(y) = \frac{\sum_{z'} D^{(2)}\big(N_{1+}(\cdot yz')\big)}{N_{1+}(\cdot\, y\,\cdot)} \; (if \; y \neq \text{^})$$

$$\gamma^{(1)}(\text{^}) = \frac{\sum_c D^{(2)}(c) N_c(\text{^}\cdot)}{c(x\cdot)}$$

where "^" is the special character denoting the beginning of a sentence. ("$" is used to denote the ending of a sentence.) Note that for trigrams (top-order) and bigrams (lower-order) we use two different sets of discounting parameters $D^{(3)}$ and $D^{(2)}$. In addition, $D^{(3)}$ is selected based on the trigram *count*, while $D^{(2)}$ is selected according to the *extended contexts* of the bigrams (except for the bigrams starting with "^", which have no "extended contexts").

$D^{(3)}$ and $D^{(2)}$ are also tuned on the HELDOUT dataset: $D_1^{(3)} = 0.52, D_2^{(3)} = 0.37, D_{3+}^{(3)} = 0.88, D_1^{(2)} = 0.55, D_2^{(2)} = 1.71, D_{3+}^{(2)} = 2.22$. After the parameters are set, TRAIN and HELDOUT are combined as training data to retrain the model.

(d)

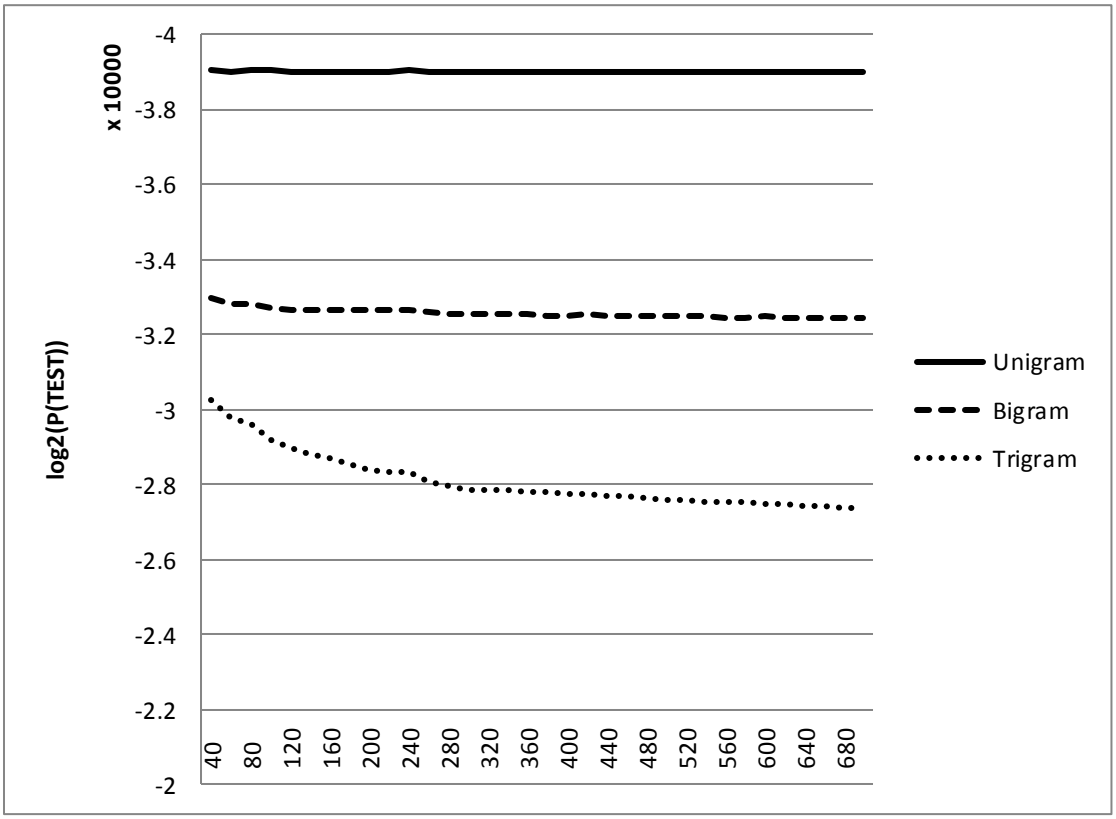
The WFSA for trigram model is generated according to the following rules:

I. "^" is the starting state, "$" is the final state
II. For each trigram "xyz" that appears in training corpus
   i. If $z \neq \$$, draw an arc from state "xy" to "yz" with transition "z" and weight $\alpha^{(3)}(z|xy)$
   ii. If $z = \$$, draw an arc from state "xy" to "$" with transition ϵ and weight $\alpha^{(3)}(\$|xy)$
III. For each bigram "xy" that appears in training corpus
   i. Draw an arc from state "xy" to "y" with transition ϵ and weight $\gamma^{(2)}(xy)$
   ii. If $y \neq \$$, draw an arc from state "x" to "xy" with transition "y" and weight $\alpha^{(2)}(y|x)$
   iii. If $y = \$$, draw an arc from state "x" to "$" with transition ϵ and weight $\alpha^{(2)}(\$|x)$
IV. For each unigram "x"
   i. If $x \neq \$$, draw an arc from state "x" to "?" with transition ϵ and wight $\gamma^{(1)}(x)$
   ii. If $x \neq$ ^ and $x \neq \$$, draw an arc from state "?" to "x" with transition "x" and weight $P^{(1)}(x)$
   iii. If $x = \$$, draw an arc from state "?" to "$" with transition ϵ and weight $P^{(1)}(\$)$

2.

The graph is shown below. For all three models, P(TEST) generally increases as more training data are provided, showing that the modeling assumptions are to some extent reasonable, in that the test data are not systematically different from the training data from the models' perspective. However, the curves of the unigram and bigram model flatten out fairly

quickly, suggesting that providing even more training data won't really help much. On the other hand, the trigram curve hasn't quite flattened out yet, so some more training data might be helpful to further improve its performance.



3.

The probability assigned to the first one is 2.95016949187e-17, to the second is 2.3159570317891e-21. The model prefers the first sequence by several magnitudes.

References

CHEN, S.F. AND GOODMAN, J., 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pp. 310–318.

JAMES, F., 2000. Modified Kneser-Ney Smoothing of n-gram Models.