All three methods perform about equally well, assuming the same amount of pruning.

The difference is that HashTBO can store more *n*-grams in the same memory and therefore it doesn't have to do as much pruning. Figure 4 shows that HashTBO consumes 3 bytes per *n*-gram whereas ZipTBO consumes 4.

Figure 4 combines unigrams, bigrams and trigrams into a single *n*-gram variable. Figure 5 drills down into this variable, distinguishing bigrams from trigrams. The axes here have been reversed so we can see that HashTBO can store more of both kinds in less space. Note that both HashTBO lines are above both ZipTBO lines.

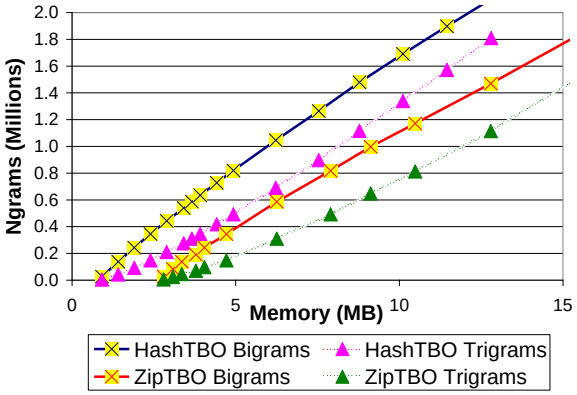

Figure 5. HashTBO stores more bigrams and trigrams than ZipTBO in less space.

In addition, note that both bigram lines are above both trigram lines (triangles). Aggressively pruned models have more bigrams than trigrams!

Linear regression on this data shows that HashTBO is no better than ZipTBO on trigrams (with the particular settings that we used), but there is a big difference on bigrams. The regressions below model *M* (memory in bytes) as a function of *bi* and *tri*, the number of bigrams and trigrams, respectively. (Unigrams are modeled as part of the intercept since all models have the same number of unigrams.)

$$M_{HashTBO} = 0.8 + 3.4bi + 2.6tri$$
$$M_{ZipTBO} = 2.6 + 4.9bi + 2.6tri$$

As a sanity check, it is reassuring that ZipTBO's coefficients of 4.9 and 2.6 are close to the true values of 5 bytes per bigram and 2.5 bytes per trigram, as reported in Section 7.3.

According to the regression, HashTBO is no better than ZipTBO for trigrams. Both models use roughly 2.6 bytes per trigram. When trigram models have relatively few trigrams, the other coefficients matter. HashTBO uses less space for bigrams (3.4 bytes/bigram << 4.9 bytes/bigram) and it has a better intercept (0.8 << 2.6).

We recommend HashTBO if space is so tight that it dominates other concerns. However, if there is plenty of space, or time is an issue, then the tradeoffs work out differently. Figure 6 shows that ZipTBO is an order of magnitude faster than HashTBO. The times are reported in microseconds per n-gram lookup on a dual Xeon PC with a 3.6 ghz clock and plenty of RAM (4GB). These times were averaged over a test set of 4 million lookups. The test process uses a cache. Turning off the cache increases the difference in lookup times.

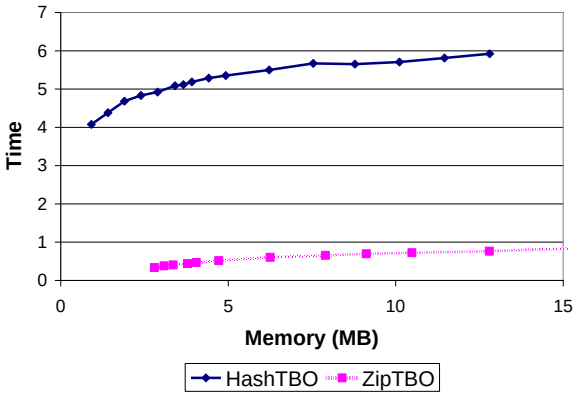

Figure 6. HashTBO is slower than ZipTBO.

9 Conclusion

Trigram language models were compressed using HashTBO, a Golomb coding method inspired by McIlroy's original spell program for Unix. McIlroy used the method to compress a dictionary of 32,000 words into a PDP-11 address space of 64k bytes. That is just 2 bytes per word!

We started with a large corpus of 6 billion words of English. With HashTBO, we could compress the trigram language model into just a couple of megabytes using about 3 bytes per n-gram (compared to 4 bytes per n-gram for the ZipTBO baseline). The proposed HashTBO method is not fast, and it is not accurate (not lossless), but it is hard to beat if space is tight, which was the case for the contextual speller in Microsoft Office 2007.