Exploring Smoothing Techniques in Language Models Used for Email Recovery

David Philipson and Nikil Viswanathan
{pdavid2, nikil}@stanford.edu
CS224N Fall 2011

Introduction

We explore different probabilistic language models for piecing together jumbled email conversations. We explore the smoothing techniques of absolute discounting, Katz backoff, and Kenyser-Ney for unigram, bigram, and trigram models. In the proceeding sections, we discuss the mathematical justifications for these smoothing techniques, present the results, and evaluate our language modeling methods. We also present our recommendation of the optimal smoothing methods to use for this application.

Approach

Initial Attempts: Absolute Discounting Unigram, Bigrams, Katz Backoff, Trigrams

We initially built a simple unigram model and continued to incrementally improve it. First we tested add-one (Laplace) smoothing and absolute discounting. Next we experimented with using bigrams and using the Katz algorithm to backoff to unigram probabilities which saw a substantial boost in Enron email performance from the unigram only language model. Building on this, we added trigrams to the model, expecting a significant increase in performance, but actually saw a slight decrease in the email task performance.

Diving Deeper: Validation Training, Kneser-Ney Bigram, Kneser-Ney Trigrams, Performance Optimizations

We used a validation set to tune the absolute discounting parameters of the Katz backoff bigram model, as it had the best performance so far. We tested on the validation set different parameter combinations for the one count discount and the multi count discount in step sizes of less than or equal to .5 unit increments. We found that values slightly below the recommended .5 and .75 discounts performed the best on the validation set measure perplexity and that when we tried it on the Enron test data we saw a word error rate improvement of about 1-2 percent.

Looking to improve our model, we considered several different approaches and researched the relative performance of language model smoothing techniques. From Chen and Goodman[1] we saw that Kneser-Ney smoothing offered the best performance (even better than linear interpolation of backoff parameters learned form a validation set) so we adopted this approach. First we implemented Kneser-Ney smoothing for bigrams using an absolute discounting smoothing for the unigram scenario. After playing around with the discount parameters we saw

[1] Stanley Chen and Joshua Goodman (1998), An empirical study of smoothing techniques for `language modeling.

a similar level performance to the smoothing from Katz backoff. The introduction of trigrams to Kneser-Ney smoothing also resulted in a similar level of performance as trigrams smoothed by the Katz backoff technique that we implemented before. In order to allow our model to train and run on trigrams and on larger data sets, we went through several rounds of optimizations, such as precomputing the Kneser-Ney “continuation probability” for all possible words in the vocabulary.

Optimal Performance: Data, Data, Data
Finally we experimented with trying out different training sets for the different algorithms. We were surprised to see the magnitude by which the source and magnitude of training data actually impacted the overall performance. Switching from the Europarl data to the Enron corpus for training, while keeping a constant size of 40,000 sentence lines, provided a moderate increase of about 5-6% for each of the models. The first increase in the data set size, a 4x increase to 160,000 lines, improved the performance of our models approximately 13%, so our best model had about 88.7% of the Enron sentences correct, and next increase in data set size, a 7.7x increase to 1,240,209 sentences (the entire remaining Enron corpus), again boosted the performance another ~10% to 98.3% correct on the Enron data jumble.

Correctness

It is easy to verify that our discounted unigram model gives probabilities which sum to 1. When we subtract D from a count, we decrease the probability mass by D / (total number of unigram tokens), and so if we give the total amount of count discounted divided by the total number of tokens as the probability for unknown tokens, the total probability mass is once again 1.

For Katz backoff bigrams, we must be sure that for any single word context, the sum of the probabilities equals 1. Our model first checks if the context has ever been seen before, and if not simply returns the unigram probability. Since we above showed that the unigram probability sums to 1, we are therefore safe in this case. On the other hand, if the context has been seen before, and this context-word bigram has been seen in the past, we return a discounted probability:

(count of this bigram - D) / (total number of bigram tokens).

This discount removes probability mass equal to D / (total number of bigram tokens). Let M be the sum of this mass over all word choices- this is probability mass which we will distribute over the words for which the context-word bigram has never been seen before. To each such word, we would like to distribute mass proportional to that word’s unigram probability, and so in the case where the context-word bigram has not been seen, we assign probability

M * (unigram probability) / (total unigram probability of words not seen in context-word bigram)

If we sum over all the words in this case, we get

M * (total unigram probability of words not seen in context-word bigram) / (ditto) = M

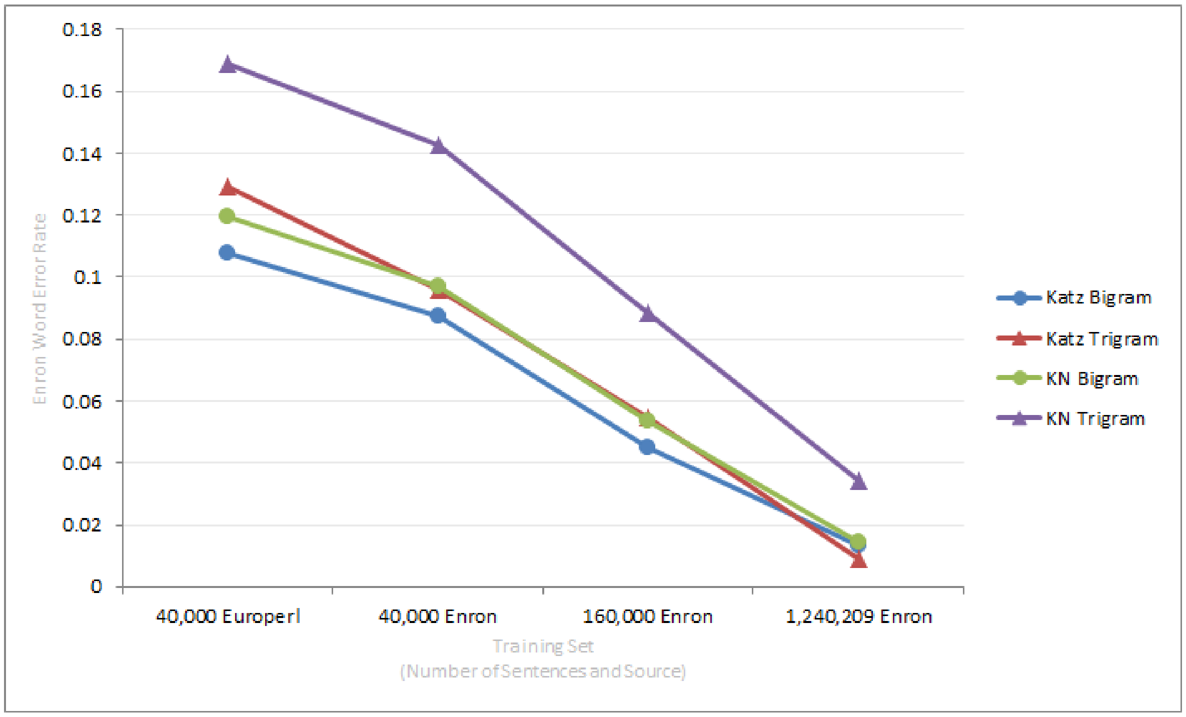
That is, we have shaved M probability mass off of the previously seen bigram case and given it to the unseen bigram case, hence the total probability is 1. To extend to the trigram case, we make the same argument, this time using the fact that we have previously verified that the Katz backoff bigram model has total probability 1 for any context.

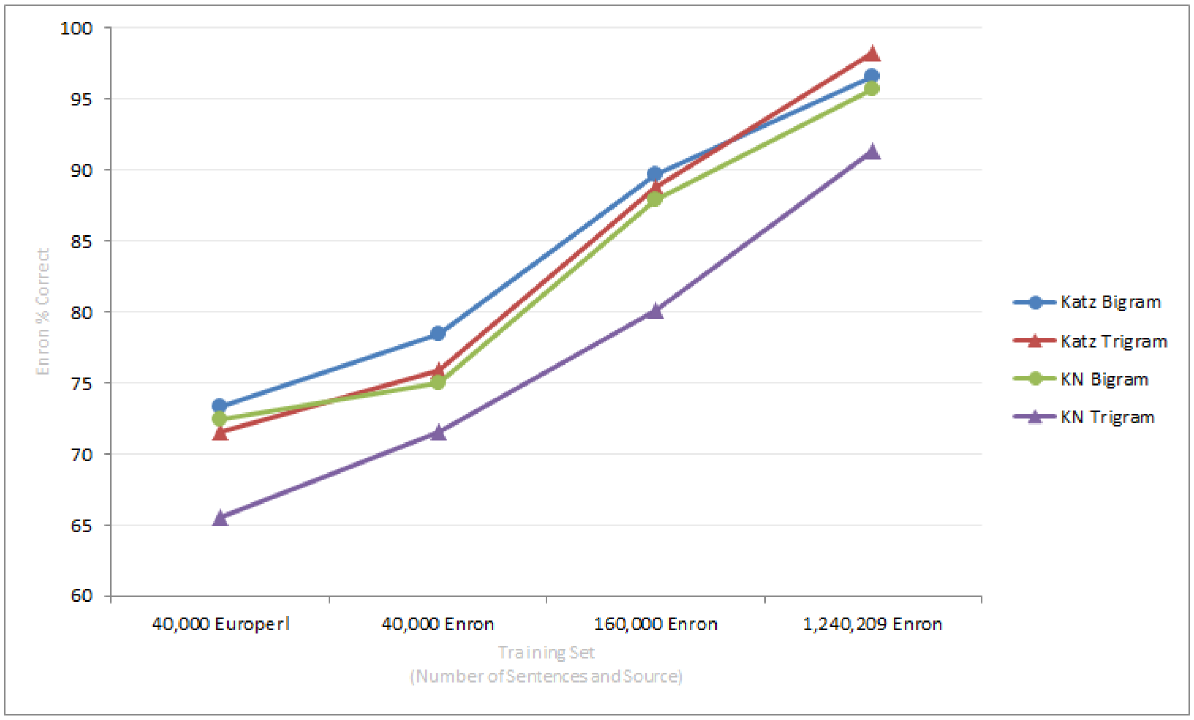
We now verify the correctness of our Kneser-Ney algorithm for n-grams. As in Katz backoff, for unseen context we back off to the Kneser-Ney model for (n-1)-grams, so by induction we are correct in the case of unseen context. For seen contexts, we start with a base discounted probability by the same formula as above. Once again, let M be the total probability mass which has been discounted. We wish to distribute this mass among all n-grams with this context proportional to how many unique n-gram types end with the given word. Hence, we add in the second probability

M * (number of n-gram types ending in this word) * (total number of n-gram types)

and once again, summing this expression over all possible words gives us M, which is exactly the amount that the base probability was decreased by. Hence, the probability sums to 1.

Results





Discussion

Our first attempt, the unigram model with absolute discounting, showed slightly worse train set perplexity than the given model, which is essentially a maximum likelihood estimate model, but about 8% lower test set perplexity. This makes sense because moving away from maximum likelihood estimates will only increase train set perplexity, but allowing a higher chance for …....unseen words gives better perplexity on other data sets. As expected, the unigram model performs no better than random selection for the task of unscrambling the Enron sentences, since it uses no information regarding the order of words in the sentence.

Naturally, moving up to a bigram model gives much better predictive powers. While training with the Europarl training set, we found that moving from Katz backoff bigrams to trigrams actually made our model perform worse at unscrambling the Enron sentences. In particular, the move from bigrams to trigrams saw a significant decrease in training set perplexity and a somewhat smaller decrease in test set perplexity, while the Enron jumble perplexity went up. These facts together imply that we were encountering overfitting: we were performing extremely well on the training set and somewhat better on a test set similar to the training set, but worse on a test set which was somewhat different. We predicted that by switching to an Enron training set we would see better performance, a hypothesis which will be confirmed below.

We implemented Kneser-Ney bigram and trigram models and found that their performance was very close to that of the corresponding Katz backoff models, and in particular the trigram model performed somewhat worse. At first we believed that this might be because we chose a poor discounting method. However, when we changed the parameters for discounting, although we saw changes to perplexity (mostly increases), changing these parameters had very little effect on the models' ability to unscramble sentences. Even extreme changes, such as discounting 0.9 or 0.1 from counts of 1 caused the scramble success rate to change by only about a percent or so.

In order to explain the worse performance of Kneser-Ney models, we examined the sentences which were coming up wrong in the scramble. At this point, we had a fairly high success rate, and many incorrect sentences would stand as reasonable sentences in their own right, for example

Correct: we have never done that before
Our guess: that we never have done before

Other sentences had amusing mix-ups, such as

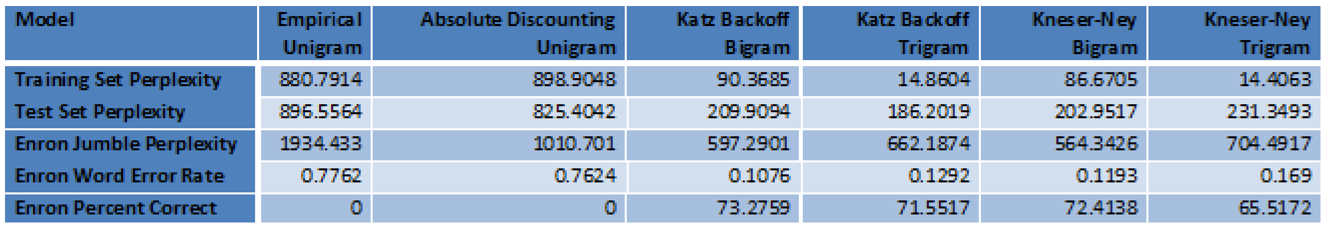
Correct: maybe we should meet in az some time in feb we can go see jan hutchens etc
Our guess: maybe we should meet in jan some time in feb we can go see az hutchens etc

in which “az” and “jan” are alternatively used as abbreviations and names. When we got a sentence really wrong, it tended to appear like
Correct ordering: a memo written by jake thomas on april 7 1999 memorializes this line of thought
Our guess: 7 1999 thomas by written of this memorializes jake a memo line on april thought

Examining this sentence, the problem seems to be that in fragments of three words or so, it seems reasonable. For example, “written of this,” “this memorializes jake” and “a memo line on april” are all perfectly reasonable pieces of a sentence. The parts that are notably wrong are “ thomas by written” and “of this memorializes,” which stand out as ungramatical. However, neither of these seem to be problems which Kneser-Ney is designed to solve: the words at the end of each of these fragments can safely appear after numerous contexts and neither would be significantly restricted by Kneser-Ney. This helps suggest why Kneser-Ney did not give us improved performance; it diverted a well-performing model in an attempt to solve a non-existent problem.

On all bigram and trigram models, we saw a significant, massive gain in correctness on the Enron scrambles by switching the training set to a large portion of the Enron corpus, rather than the default small Europarl training set. Indeed, by simply increasing the training set size from 160k to 1200k sentences, we saw about a 10% increase in correctness from the 80-88% range to the 90-98% range. Furthermore, by switching to Enron training sets from Europarl sets, the Katz backoff trigram model overtook the Katz backoff bigram model, becoming our most successful model and confirming our above prediction. Indeed, with a 98.3% correctness rate and a 0.88% word error rate on the Enron scrambles, we believe we have trained a successful model for the task.

Data



Model	Empirical Unigram	Absolute Discounting Unigram	Katz Backoff Bigram	Katz Backoff Trigram	Kneser-Ney Bigram	Kneser-Ney Trigram
Training Set Perplexity	880.7914	898.9048	90.3685	14.8604	86.6705	14.4063
Test Set Perplexity	896.5564	825.4042	209.9094	186.2019	202.9517	231.3493
Enron Jumble Perplexity	1934.433	1010.701	597.2901	662.1874	564.3426	704.4917
Enron Word Error Rate	0.7762	0.7624	0.1076	0.1292	0.1193	0.169
Enron Percent Correct	0	0	73.2759	71.5517	72.4138	65.5172

	Katz Backoff Bigram	Katz Backoff Trigram	Kneser-Ney Bigram	Kneser-Ney Trigram
Europarl 40,000 Lines				
Test Set Perplexity	209.9094	186.2019	202.9517	231.3493
Enron Jumble Perplexity	597.2901	662.1874	564.3426	704.4917
Enron Word Error Rate	0.1076	0.1292	0.1193	0.169
Enron Percent Correct	73.2759	71.5517	72.4138	65.5172
Enron 40,000 Lines				
Test Set Perplexity	711.4787	847.1166	673.9393	852.957
Enron Jumble Perplexity	473.3289	539.2155	464.7388	613.6043
Enron Word Error Rate	0.0875	0.0961	0.0968	0.1428
Enron Percent Correct	78.4483	75.8621	75	71.5517
Enron 160,000 Lines				
Test Set Perplexity	738.25	842.5508	705.5968	899.9417
Enron Jumble Perplexity	412.7423	431.1092	408.3787	492.055
Enron Word Error Rate	0.045	0.0548	0.0538	0.0886
Enron Percent Correct	89.6552	88.7931	87.931	80.1724
Enron 1240209 Lines				
Test Set Perplexity	726.8178	784.1253	698.1782	877.4087
Enron Jumble Perplexity	240.6183	91.0171	234.2358	102.8218
Enron Word Error Rate	0.0132	0.0088	0.0144	0.0345
Enron Percent Correct	96.5517	98.2759	95.6897	91.3793