Write Up for Algorithms for NLP: 11711 Assignment 1

Steven Hillis
Language Technologies Institute, Carnegie Mellon University

Abstract

This paper deals with the implementation of the Kneser-Ney Smoothing algorithm for the trigram language model. The implementation is constrained by runtime and memory usage requirements, and it is evaluated based on the downstream task of using machine translation to decode French sentences to English ones.

1 Introduction

A language model is used to predict the probability of a word given its occurrences in a training data set. It necessitates a Markov assumption, indicating that the probability of a word can be determined based on a certain quantity of past information. In a unigram language model, predictions are made based on the probability of the unigram occurring in the training corpus. This model predictably underperforms, as it makes no attempt to account for the context of the word being predicted.

One solution to this lack of context is a trigram language model, which takes into account the context in which the word we are predicting appears. A trigram model performs much better than a unigram model, but like all simple n-gram models, it fails to predict unseen words at test time. In order to predict unseen words at test time, the probability mass of the distribution is smoothed, reserving a measure of likelihood for words not seen during training.

The naïve approach to smoothing is known as Laplace smoothing, and it involves adding 1 to the count of every word (Chen and Goodman 1998). This approach is useful in many domains, but it lacks luster for a language modeling task where the number of unobserved outcomes is so large.

There are many other, more appropriate smoothing approaches for language modeling. Contemporary methods focus on time and storage efficiency, and they include storing implicit randomized representations of n-gram statistics (Talbot and Osborne 2007), perfectly hashing n-grams together with their probabilities using the Bloomier filter (Talbot and Brants 2008), and direct address caching (Pauls and Klein 2011).

Our approach, proposed by Kneser and Ney, involves a combination of discounting and backing off in order to optimize available information and account for contexts and utterances unseen during training (1995).

2 Implementation Details

The implementation of the trigram language model with Kneser-Ney Smoothing was bounded by certain constraints. The language model itself was constrained to around 900M of memory, and the JVM was constrained to 2G of memory during the training of the model. Decoding time itself was limited to be 50x slower than the time taken to decode the unigram model, and the total time to build and decode using the trigram model was limited to 30 minutes. Finally, the BLEU score resulting from the use of the model in a machine translation task was given a lower bound of 23.

The model described in this paper was implemented with an open-address hashing strategy. Our hash tables were instantiated with prime valued sizes and were rehashed after reaching a threshold of eighty-five percent of the total size of the table. Keys were reported modulo the size of the table. We adopted quadratic polling—after a collision, we polled the table at quadratically increasing distances from the previous polling index—for its advantages in speed over linear polling. When the size of the table reached the threshold, we rehashed the table as a whole and increased the total table size by the prime number above 100% of the previous table size.

Trigrams and bigrams were encoded in the model using a bit packing method, so that ngram arrays of primitive integers could be condensed into primitive longs. Unigrams were encoded simply as the index to arrays of primitive integers storing the counts of the unigram, of its unique contexts, of its unique trailing contexts, and of its unique sandwiching trigrams. Counts of trigrams were stored as integers. For bigrams, contexts and

trailing-contexts were stored along with counts as integers in a multi-valued implementation of our hash map.

When calculating the probability of an unseen word in in unseen context, a constant equal to the proportion of 1 to the vocabulary size was returned. Otherwise, probabilities were computed according to the Kneser-Ney Smoothing algorithm.

3 Empirical Results

Our model failed to meet the standards of the assignment. It was unable to build the full model using the prescribed maximum heap size of 2000M.

In order to allow our model to run as far as it was able given its inefficient use of memory, we set our load factor for our hash maps at 0.85. This decision cost us in build speed, but the trade off was necessary to allow more memory for the model. To accommodate for this atypically high load factor, we initialized our trigram and bigram hash tables to large (10^7 magnitude) prime numbers.

3.1 Results Compared to Empirical Unigram Model

Model	BLEU score	Decoding Time	Memory Usage
Unigram	9.253	10.9	138M
Trigram with KN Smoothing	21.819	332.2	627M

Table 1: Results for Unigram vs Trigram Models on 250,000 sentence size training data

The Trigram model with KN Smoothing is clearly superior in performance on the downstream task, but it also requires more memory and time.

3.2 Results for Varied Sizes of Training Data

Number of training sentences	10,000	100,000	250,000
Vocabulary Size in training data	16055	39585	59222
Number of Trigrams	164085	916626	1913021
BLEU score	18.180	20.529	21.819
Decoding Time (s)	204.2	251.4	332.2
Memory Usage	164M	182M	627M

Table 2: Results for Trigram Model with KN Smoothing for Various Training Data Sizes

As with the transition from Unigram to Trigram data models, increasing the size of the training data sets increases performance at the cost of memory and time.

4 Error Analysis

The failure of our model illustrates a pressing contemporary issue in the field: balancing training data size with resource consumption. As incomprehensible amounts of data have become freely available (Goldberg and Orwant 2013), balancing performance (given that performance is most often proportional to training data size in language models) with available resources has become more and more necessary. Here, the implementation of our models bounds us by an even stricter limitation on resource consumption than designed for the problem. We are left to assume that training the model on the full corpus of approximately 9 million sentences would continue the trend in BLEU score improvements.

4.1 Machine Translation Errors

We now compare the translation output for a single sentence using the four different model sizes from 3.2.

Input:
il n' y a pas de raison que le service offert aux populations n' intègre pas l' ensemble des progrès techniques et des progrès économiques et je crois que le fait d' avoir supprimé les services spéciaux est , de ce point de vue-là , une avancée avant une clarification .

Hypothesis with 10k model:
it n' has no reason that the service of the people n' integrates not l' together progress and technical progress economic and i believe that the fact d' be deleted services special is , in this respect , a progress before a clarification .

Hypothesis with 100k model:

it n' there was no reason that the service provided on the populations n' integrate l' together progress technical and economic progress and i believe that the fact d' having abolished the special services is , of this point of view , a progress before a clarification .

Hypothesis with 250k model:
it n' there was no reason that the service provided for the population n' integrate l' range of technical progress and economic progress and i believe that the fact d' having abolished the services of special , of this point of view , there is a breakthrough before a clarification .

Reference:
there is no reason why the service offered to people should not incorporate all kinds of technical and economic progress , and i believe that , from this point of view , doing away with special services is a step for which no clarification had first been provided .

Clearly, fluency improves with the size of the model's training data. It is interesting to note that the French articles are included in all three english translations; perhaps given a large training set, these utterances could be accommodated in the translation model.

5 References

Adam Pauls and Dan Klein. 2011. Faster and smaller n-gram language models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics

David Talbot and Miles Osborne. 2007. Smoothed Bloom Filter Language Models: Tera-scale LMS on the Cheap. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).*

David Talbot and Thorsten Brants. 2008. Randomized language models via perfect hash functions. In *Proceedings of ACL-08: HLT*, 505-513.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *icassp* Vol. 1.

Stanley F. Chen and Joshua Goodman. 1998. *An Empirical Study of Smoothing Techniques for Language Modeling.* Center for Research in Computing Technology, Harvard University, Cambridge, Massachusetts.

Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic ngrams over time from a very large corpus of english books. *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*. Vol. 1.