Language Modelling with SRILM

Speech Technology

Chi NGUYEN, Quan NGUYEN, Cuong NGUYEN

University of Hamburg, Department of Computer Science, Germany

Abstract

This paper presents the use of SRILM toolkit for training language models with N-grams. The toolkit provides several different language models for estimation procedures. Experiments with these language models and the results will indicate which model may perform better than the others.

Introduction

SRILM - The SRI Language Modeling Toolkit, developed by the SRI International organization, is used for building and applying statistical language models (LMs), primarily for speech recognition, and machine translation [2]. SRILM provides features for training LMs and testing them on data. Using such features, our experiments include training multiple LMs and comparing their perplexity on testing data, in order to find out which LM has the best performance in general.

Language Modeling

The goal of a language model is to predict the most likely word to follow, given a sequence of words. The idea of word prediction is formalized as probabilistic models called ***N-gram*** models, which predict the next word from the previous $N-1$ words. Such statistical models of word sequences are also called language models.

Experimental Setup

The experiment setup consists of the SRILM toolkit, the training sets and test sets generated from the Europarl corpus.

- The Europarl Corpus is a corpus that consists of the proceedings of the European Parliament from 1996 to 2006 [1]. The complete corpus covers eleven official languages of the European Union. It is usually used for statistical translation models but in our experiments, we only employ the English corpus. The corpus used in the experiments comprised of $34,571,768$ English words, which is tokenized and shuffled randomly.
- The training sets are generated from the corpus. There are eight training sets with the following coverage: $1\%, 2\%, 4\%, 8\%, 16\%, 32\%, 64\%$ and 99% of the corpus.
- The test sets are randomly generated from 1% of the corpus. There are 15 test sets which are tested against the LMs. The average perplexity measure is taken as the result.
- The LMs used for training are: simple smoothed N-grams, Witten-Bell Backoff/Interpolate, Absolute Backoff/Interpolate, Chen & Goodman's Kneser-Ney Backoff/Interpolate, and Maximum Entropy.

Perplexity

Cross-entropy of each test sentence $P_{LM}(w_1, ..., w_n)$ is computed as

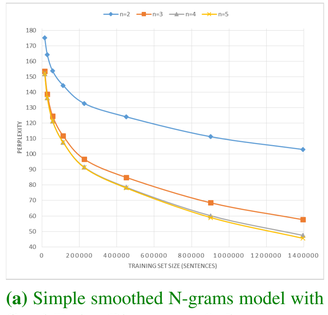
$$\begin{aligned} H(P_{LM}) &= -log(P_{LM}(w_1, ..., w_n))/n \\ &= -\sum_{i=1..n} log(P_{LM}(W_i|w_1, ..., w_{i-1}))/n \end{aligned} \quad (1)$$

The perplexity of a LM for a sentence is $2^{H(P_{LM})}$.

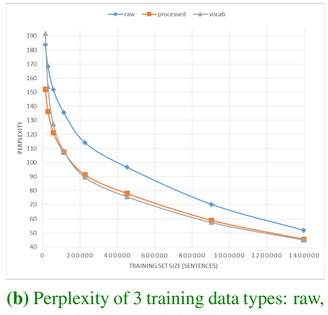
Simple N-grams

Good-Turing Discount is the default smoothing technique. Cut-off steps are as followed: unigram (1,1) bigram and further (2,7). This is said to ensure the maximum likelihood of words.

```
./ngram-count -text Train1.txt -order 5 -lm Train1.count
./ngram -lm Train1.count -ppl Test.txt -debug 2
```



(a) Simple smoothed N-grams model with Good-Turing Discount, n=2..5



(b) Perplexity of 3 training data types: raw, processed, processed-closed-vocab, n=5

Figure 1

Witten-Bell

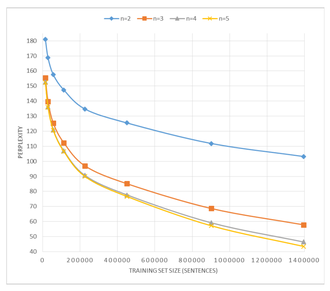
Witten-Bell method theorizes the discount factor in Simple N-grams by scaling the original Maximum Likelihood of high-order N-gram and adding the likelihood of low-order N-grams.

Backoff

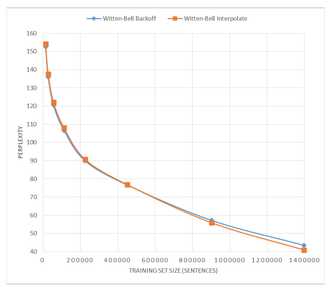
```
./ngram-count -text Train1.txt -order 5 -wbdiscount -lm Train1.count
```

Interpolate

```
./ngram-count -text Train1.txt -order 5 -wbdiscount -interpolate -lm Train
  1.count
```

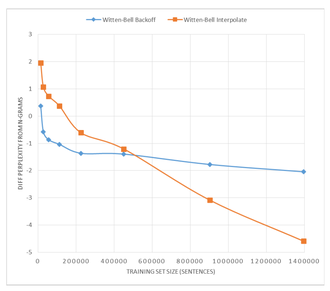


(a) Witten-Bell Backoff model, n=2..5

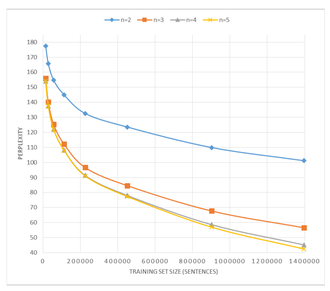


(b) Witten-Bell Backoff vs. Interpolate, n=5

Figure 2



(a) Diff Perplexity from N-grams, n=5



(b) Absolute Backoff model, n=2..5

Figure 3

Absolute Discounting

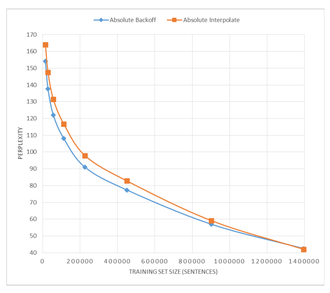
Absolute Discounting (AD) theorizes the discount factor of all observed N-grams in Good-Turing method by discounting a fixed portion from observed sequences. AD model simply backoff to the grounding case (unigram) without taking into account the preceding context.

Backoff

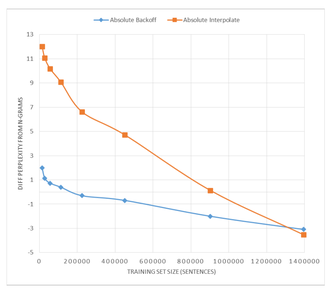
```
./ngram-count -text Train1.txt -order 5 -cdiscount 0.5 -lm Train1.count
```

Interpolate

```
./ngram-count -text Train1.txt -order 5 -cdiscount 0.5 -interpolate -lm
  Train1.count
```



(a) Absolute Backoff vs. Interpolate, n=5



(b) Diff Perplexity from N-grams, n=5

Figure 4

Chen & Goodman's Kneser-Ney

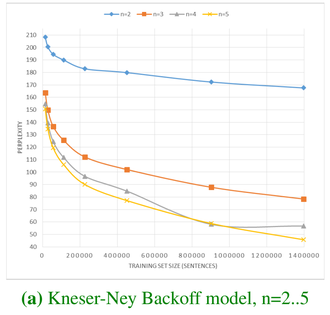
Kneser-Ney method is similar to AD method, except that the probability of grounding case (unigram) is measured as a continuation in all observed context.

Backoff

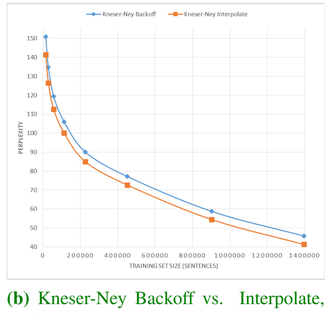
```
./ngram-count -text Train1.txt -order 5 -kndiscount -lm Train1.count
```

Interpolate

```
./ngram-count -text Train1.txt -order 5 -kndiscount -interpolate -lm Train
  1.count
```

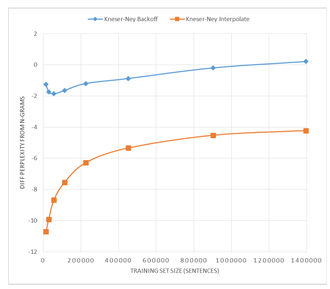


(a) Kneser-Ney Backoff model, n=2..5

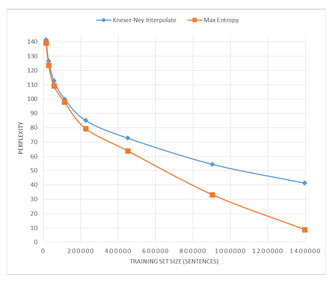


(b) Kneser-Ney Backoff vs. Interpolate, n=5

Figure 5



(a) Diff Perplexity from N-grams, n=5



(b) Kneser-Ney Interpolate vs. MaxEnt

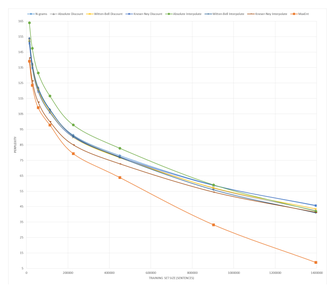
Figure 6

Maximum Entropy

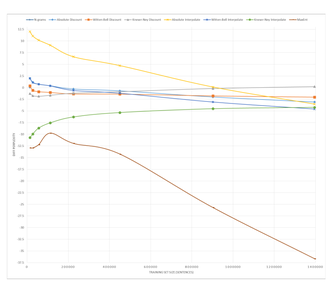
The likelihood of unseen sequences in MaxEnt method are estimated by building a probabilistic model taking into account the least assumptions and the most significant empirical observations.

```
./ngram-count -text Train1.txt -maxent -lm Train1.count
```

General Result



(a) Perplexity of all LMs, n=5



(b) Diff Perplexity from N-grams for all LMs

Figure 7

Evaluation

All measurements are taken as the average values of 15 experiments to ensure statistical significance of comparisons. In general, the performance of interpolated versions of all models surpass their backoff counterparts as a result of bringing more contextual information to the smoothing processes. Evaluations are carried out based on the following criteria:

a) Raw and Processed Data: Processed data enabled us to achieve much better performance as illustrated in Figure 1(b).

b) Open and Closed Vocab: the experiment result in Figure 1(b) suggests that the closed vocabulary setting achieves slightly better performance as the training size increases. However, the method by which the vocabulary is chosen or generated will have great effect on the performance of the model.

c) Different language models: the discriminative model Maximum Entropy outperforms all other generative methods. As training size increases, the superiority of MaxEnt method becomes even more significant compared to the most effective generative model - Kneser-Ney. Among generative models, the Witten-Bell performs better than Absolute Discounting but worse than Kneser-Ney model, which implies that handling the grounding case in recursive relation in backoff phases has a great impact on the model's performance, although the differences are not quite strong.

Conclusion

In this work, we have measured the performance of language models primarily through the perplexity. Future works include investigating the effect of part-of-speech on performance of N-grams models as well as extracting other features to improve both the Bayesian network in generative models and the feature combinations in Maximum Entropy models.

References

[1] European Parliament Proceedings Parallel Corpus. `http://www.statmt.org/europarl/archives.html#v3`.

[2] SRILM - the SRI language modeling toolkit. `http://www.speech.sri.com/projects/srilm/`.

[3] Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359 – 394, 1999.