Large Language Models in Machine Translation

Thorston Brants, et al.

Joel Robert Adams

Center for Spoken Language Understanding
Oregon Health and Science University

CS506, 2014

Outline

Machine Translation

Machine translation is a process of producing the best English sentence $\hat{e}$ given source language sentence **f**

- $\hat{e} = \arg\max_e \sum_{m=1}^{M} \lambda_m h_m(\mathbf{e}, \mathbf{f})$[1]
- If $h(\mathbf{e}, \mathbf{f}) = h(\mathbf{e})$ then it's a *language model*
- Extension of noisy channel model: $\hat{e} = \arg\max_e p(f|e)p(e)$

[1] Och and Ney, 2004

N-gram Language Models

Language models are statistical models of word sequences.

- Let $w_1^L = (w_1, ..., w_L)$ then $P(w_1^L) \approx \prod_{i=1}^{L} P(w_i | w_{i-n+1}^{i-1})$
- Maximum Likelihood estimate: $\hat{P}(w_i|h_i) = \frac{c(h_i w_i)}{c(h_i)}$
- But what about unseen words? Unseen histories? Unseen word-history pairs? What about "parasitic delicious hamburger?"
- Smoothing!

Smoothing

Smoothing is a way of reserving probability mass for unobserved n-grams. Typically they follow the pattern:

- $P(w|h) = \begin{cases} p(w|h) & \text{if } c(hw) > 0 \\ \lambda_h P(w|h') & \text{otherwise} \end{cases}$
- $p(w|h)$ is an estimate that sets aside some probability mass and λ_h ensures normalization.
- Recursive down to zero-grams

Stupid Backoff

Stupid Backoff is similar but uses frequencies rather than normalized probabilities.

- $S(w|h) = \begin{cases} \frac{c(hw)}{c(h)} & \text{if } c(hw) > 0 \\ \alpha S(w|h') & \text{otherwise} \end{cases}$[2]
- Recursive down to unigrams:

$$S(w) = \frac{c(w)}{N}$$

- What about unseen words?

[2] α set to .4 for all experiments.

Map Reduce

Training is done as a series of 3 steps.

- Vocabulary Generation: maps terms to integer IDs along with frequency counts.
- n-Gram Generation: Words are now IDs and n-grams are emitted with counts.
- Language Model Generation: n-grams and counts are transformed into relative frequencies for stupid backoff.[3]

[3] Some additional steps required for more sophisticated smoothing (Kneser-Ney).

Vocabulary Generation

On input (**key, value**), where **key** is arbitrary and **value** is some string from the training corpus:

- Map:
  - Associates an intermediate integer ID key with each token.
  - Emits these keys with their frequency in **value**.
- Reduce:
  - Sums all values for a given key.
  - Emits these sums.

Words which occur below some threshold are all mapped to a special ID representing “the unknown word.”

n-Gram Generation

Similar to Vocabulary generation. On input (**key, value**), where **key** is arbitrary and **value** is some string from the training corpus.

- Map:
  - Converts all tokens in **value** to their IDs.
  - Emits these IDs with a value of 1 (or sums their frequency in **value**).
- Reduce: Same as previous.

In order to ensure shards have frequency data for numerator and denominator, sharding is done by hashing on the first two words of an ngram.

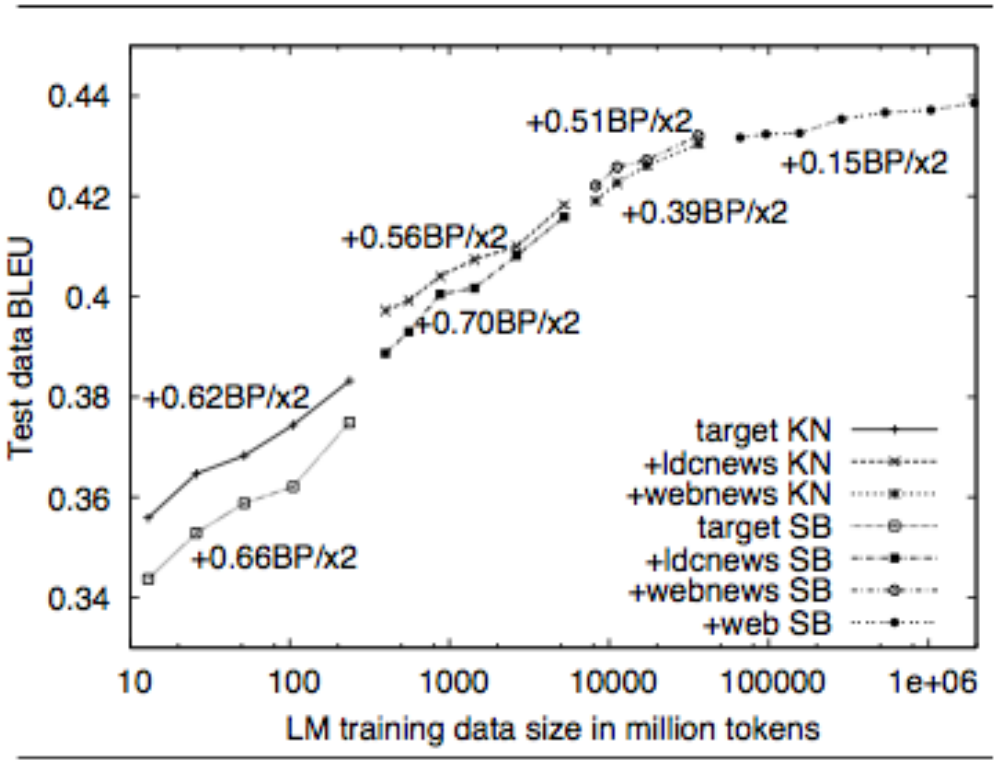
Language Model Generation

- A given shard now has ngram and count data for all words and their ngram histories. Relative frequency is calculated.
- In order to ensure that shards have required data for performing back off, sharing is done by hashing in the /textitlast two words of an ngram.

5-gram language models were trained on four training data sets.

- Training
  - **target:** English side of Arabic-English parallel data from LDC (237 million tokens)
  - **ldcnews:** Several English news data sets from LDC (5 billion tokens)
  - **web news:** Data collect for several years from English news articles (31 billion tokens)
  - **web:** General web data (2 trillion tokens)
- Test
  - NIST portion of Arabic-English NIST MT evaluation set.

Results


0.44
0.42
0.4
0.38
0.36
0.34
Test data BLEU
+0.51BP/x2
+0.15BP/x2
+0.39BP/x2
+0.56BP/x2
+0.70BP/x2
+0.62BP/x2
+0.66BP/x2
target KN
+ldcnews KN
+webnews KN
target SB
+ldcnews SB
+webnews SB
+web SB
10
100
1000
10000
100000
1e+06
LM training data size in million tokens

Training Times

	target	*webnews*	*web*
# tokens	237M	31G	1.8T
vocab size	200k	5M	16M
# n-grams	257M	21G	300G
LM size (SB)	2G	89G	1.8T
time (SB)	20 min	8 hours	1 day
time (KN)	2.5 hours	2 days	–
# machines	100	400	1500

Summary

- Training speeds really are remarkable.
- Stupid Backoff is increasingly competitive with Kneser-Ney as training data size increases.
- BLEU score still increasing.

Discussion

- Vocabulary step is separate step from n-gram creation. Could it's output be used as unigrams?
- "We assign IDs according to term frequency, with frequent terms receiving small IDs for efficient variable-length encoding."
- Optimization of doing token summing in the map step.
- "sharding function determines which shard (chunk of data in the MapReduce framework) the pair is sent to." Is this control of reducers or just storage of output of system? Is this simply a combiner?
- Non map-reduce steps in the system: Calculating N? Calculating Relative Frequencies? Remapping of low frequency words to UNK?
- General applicability of Stupid Backoff. Do cases exist (even outside of machine learning) where the use of scores rather than normalized probabilities would be a problem?
- Vaguely curious about number of shards and impact on performance.