KenLM - Fun with Language Models

Kenneth Heafield
Junfei Guo
Marcin Junczys-Dowmunt
Dmitry Khristich

CMPH Language Model

Huge LM

Language models memorize strings in the training data:
Entries from 1.8 trillion tokens

Length	Unique Strings
1	2,640,258,088
2	15,297,753,348
3	61,858,786,129
4	156,775,272,110
5	263,690,452,834

Store a model with 500,262,522,509 entries
Currently, online decoding requires 5.5 TB RAM (2.4 TB with quantized Q-values)

Saving Memory - Minimal Perfect Hashing

- CMPH - CHD algorithm
- Minimal perfect hash function
- Great: CMPH-Library, CHD algorithm: 2.06 bits per key
- Bad: Assigns a value from 1 .. N for unseen keys
- Good-enough solution: Fingerprinting using n bits from random hash function e.g. MurmurHash

Saving Memory - Sharding

- Hash the n-gram
- MurmurHash - 64bit or 128bit random hash value, use last $m + n$ bits
- Use m bits for sharding (m - command line option, there will be 2^m shards)
- Store next n bits for fingerprinting

Saving Memory – Q-Values

Unigrams		
Words	**log** p	**log** b
iran	-3.9	-0.6
is	-2.6	-1.5
one	-3.4	-1.0
of	-2.5	-1.1

Compress

Unigrams	
Words	**log** q
iran	-4.5
is	-4.1
one	-4.4
of	-3.6

Fewer values to remember $\implies$ 11–26% reduction in RAM usage.

Projected size-reduction

Status: we programmed a little bit, but we now know how to do it.

Bits	FP Ratio	Size (TB)
10	1:1024	1.25
12	1:4096	1.38
16	1:65536	1.63

Together with aggressive pruning might actually fit into RAM of an affordable machine

Class-based Language Models with Modified Kneser-Ney Smoothing

Discount Injection for Class-based models

- Modified Kneser-Ney Smoothing requires the presence of n-grams with counts 1, 2, 3 to calculate discounts
- Problem: unigrams (and lower order n-grams) in class-based models occur hundreds of times, there may be no n-grams with counts 1, 2, 3
- Solution: Injecting fall-back discounts where this failed.

Results

Status: It's alive!

Baseline:

- 18M sentences, English-Spanish, UN resolutions
- Vanilla Moses, language model trained on target training data.
- Word Cluster IDs calculated with word2vec, 200 clusters

System	BLEU
Baseline	58.43
+WC-LM (IRSTLM, Witten Bell)	59.84
+WC-LM (KenLM, MKN)*	60.25

* Using weights that have been tuned with the IRSTLM model.

Tunable Discounts

Modified Kneser-Ney Discount

- Chen and Goodman (1996) replace the only one Kneser-Ney-discount by the discount function fixed on the training data
- However, still mismatch between training data and test data
- Especially in the case: the training data domain is different from test data domain

Proposed Methods

- MITLM: Iterative Language Model Estimation by tuning KN-discount parameters to minimize Development set perplexity with Powell's method
- Polynomial Discount Method: POLKN class-based model with polynomial discounting, optimize parameters on development set

  Replace the KN-discount D by the discounting function $E(c) = \rho \cdot c^{\gamma}$

KenLM

Status: Research in progress...

- Implement one of the discount models in KenLM
- Tune the parameters on development set