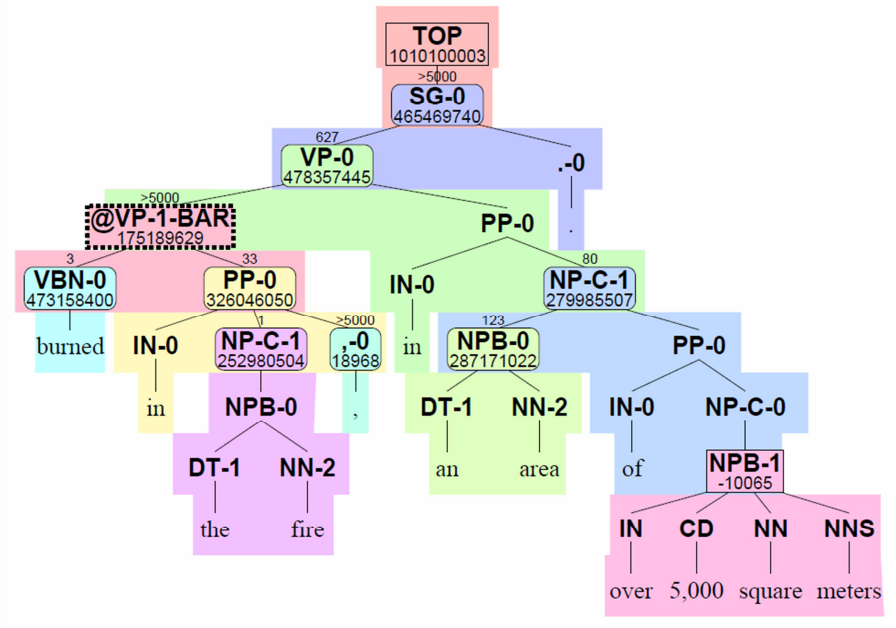
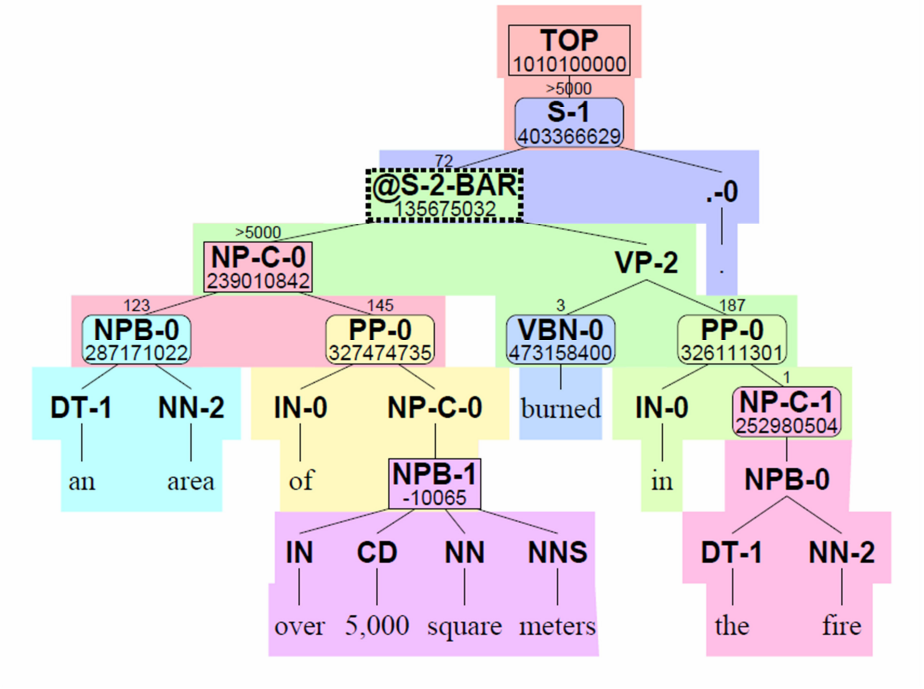
WHAT MT CURRENTLY DOES (each translation rule is a colored polygon):

在这场火灾中，过火面积达 5000 多平方米。 →



WHAT WE WISH IT WOULD DO (force decoding for Kevin-edited string – see https://groups.google.com/group/isi-mt for more):

在这场火灾中，过火面积达 5000 多平方米。 →



Both outputs are in the search space of the decoder. Unfortunately, the first output scores better than the second. What feature(s) could we add to the system that would make it prefer the second? How about a feature that **rewards well-formed English trees**?

Kevin's earlier handout (November 2010) described possible tree-based language models and experiments.

Here we'll consider the simplest form of tree-based language model, a PCFG. The PCFG rewards trees that have commonly-seen CFG rewrites. Compare e.g. the force-decode single level CFG rewrite "S→NP-C VP .", which is better than SBMT's "SG→VP ." Note that the syntax in the second tree isn't perfect; the VBN should be VBD.

Why would MT output trees with bad CFG rewrites?

Note that our baseline also includes a lexicalized dependency language model, which may indirectly encourage better trees already. Also, the CFG rewrites within a rule (same-colored polygon) are reasonable; we use P(rule) ~ count(rule) and so should have a joint generative PCFG+ (memorizing larger chunks). So why do we see any bad rewrites at all? Consider this example:

Original Penn Treebank style fragments:
S(NP ADV VP)
S(NP VP PP)
re-structured Etree fragments:
S(NP @S-BAR(ADV PP))
S(@S-BAR(NP VP) PP)
learned rewrites:
S → NP @S-BAR, SBAR => ADV PP
S → @S-BAR PP, **@S-BAR → NP VP**
bad tree created from learned rewrites:
S(NP @S-BAR(NP VP)) "two subjects"

The existence of @[NT]-BAR nodes means that we can produce novel CFG rewrites in decoding (good), but we're also unhappy with many of them (bad).

PCFG training

If it weren't for tree binarization, a PCFG feature could just be a constant precomputed (log) probability for each rule's rewrites. In this work, I score original Treebank CFG rewrites, which can cross multiple SBMT rules.

I trained a PCFG on the English trees from the Treebank parses of the bitext (2.8M sentences). For smoothing, I used an almost-ngram left to right backoff that also includes the parent as context;

e.g.:

p(S→NP DT) = p(<s> [S] → NP)*p(<s> NP [S] →DT) * p(<s> NP DT [S] → </s>).

This allows using ngram tools for training, backoff computation and scoring. Words are always generated by preterminals e.g. DT("the"), so the resulting model is small (800k 5grams, or 15MB), and can be trained with my Python code or the SRI toolkit.

Our MT system generates a few "unknown" (not in training data) tags (e.g. GLUE for fallback monotone assembly of phrases) and words (e.g. from rule-based named-entity rules). These get a tuned probability by way of indicator features counting the number of unknowns.

PCFG evaluation

model	Backoff weight method	Test log2prob/node
Qing PCFG+unigram BO	Witten-Bell	3.25
Steve PCFG+unigram BO	Witten-Bell	3.23
Steve PCFG+unigram BO	Good-Turing	3.22
My 4-gram p(c3|c1 c2 P)	Witten-Bell	3.18
My 5-gram	Witten-Bell	3.17
My 6-gram	Witten-Bell	3.17

I didn't yet investigate Good-Turing or mod-KN, since my histories aren't normal ngram ones.

SBMT decoder feature for integrated PCFG scoring

I modified the decoder to efficiently score the PCFG model as soon as possible, so that the resulting forests and nbests include the PCFG (log prob, #unknowns, and #rewrites) features. To do this, I needed each hypothesis with a restructured-label (@-xxx-BAR) as its etree-root to be distinguished by the list of Treebank children. Since this could cause search error, I added a greedy option, which keeps only the best scoring list of children for that item (having the same ngram state, category, span, etc.).

Greedy PCFG scoring turns out to be unnecessary. This is unlike the dependency LM feature, which needed greedy scoring to be practical.

I only scored syntax rules (in theory some large syntax rules could be partially scored before completion – mostly composed rules).

Sibling bigram and parent:child indicator features

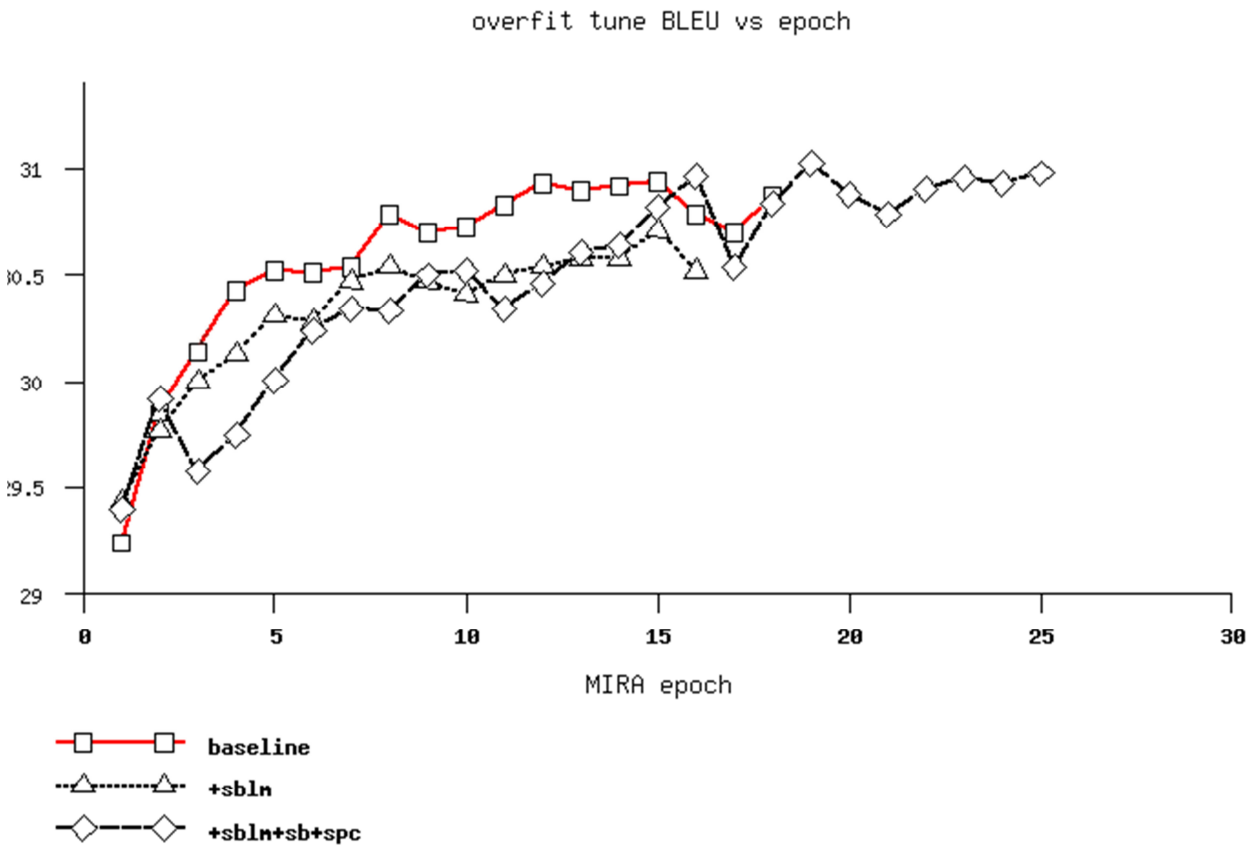
Supplementing the PCFG probability+size+unknowns indicators:

- 905 adjacent-nonterminal bigram features e.g. sb[NP-C][VP]=2
- 447 parent-child bigram features e.g. spc[TOP][SG]=1 and spc[NPB][DT]=3. Supplements existing probabilistic P(TOP→xxx) feature; we were dissatisfied in inspecting system output with how often we got "weird" sentence roots like SBAR-Q
- Both may cross rule boundaries due to @NT-BAR tree-restructured labels as rule LHS roots

BLEU went nowhere

Chinese 8.1 gale-baseline, tuned on tune (3000 sentences), evaluated on SCT (3000 sentences).

Configuration	Tune bleu	SCT bleu [length ratio]
Baseline	30.7	29.7 [0.99]
Baseline+pcfg (no p(word|tag) – NTs only)	30.7	29.6 [1.00]
Baseline+pcfg+sb[DT][NN]+spc[NPB][DT] (no p(word|tag) – NTs only)	31.0	29.7 [1.00]



I tried having a separately-weighted p(word|tag) feature; above is with no p(word|tag) at all.

When you add useless or bad features to a strong baseline, then tune, then decode, you should expect slightly higher scores on tune (more overfitting), and perhaps slightly worse scores on held-out. My observations were consistent with this unflattering null hypothesis:

Does anything work?

We know that the PCFG model is correct because its trained using standard ngram tools and its probs sum to 1, while its perplexity is similar to others' implementations.

I know that the indicators and probabilities computed during decoding are correct by verification using a separate Python nbest scorer, which scores the etree as a whole, rather than incrementally at each syntax rule.

We can see that the newly added features favor human-postedited (ok, Kevin-postedited) MT outputs' best derivations' syntax over baseline MT output:

(feature higher in MT than in postedit)	feature lower in MT than in postedit)
sblm delta=-2.777181 N=46	spc[VP][PP] delta=0.6 N=10
sblm-nts delta=-0.7608696 N=46	sb[PP][</s>] delta=0.2 N=30
sb[NPB][</s>] delta=-0.1086957 N=46	sb[<s>][VB] delta=0.5833333 N=12
sblm-unkcat delta=-0.06521739 N=46	sb[<s>][VP] delta=0.7 N=10
sb[<s>][JJ] delta=-0.1052632 N=19	sb[VP-C][</s>] delta=0.3888889 N=18
spc[ADJP][RB] delta=-1 N=1	sb[<s>][VBZ] delta=0.4444444 N=18
sb[<s>][NNP] delta=-0.0625 N=16	spc[VP][VBZ] delta=0.4444444 N=18
spc[NP-C][NPB] delta=-0.02173913 N=46	sb[NPB][PP] delta=0.4705882 N=17
spc[ADVP][RB] delta=-0.08333333 N=12	sb[<s>][IN] delta=0.2105263 N=38
…	spc[NPB][NNP] delta=0.32 N=25

We can also see that the gradient in baseline MT's nbest list already favors (on average) better PCFG probabilities, without having that as an explicit goal:

(average change in baseline MT from 1st-best to 2nd-best, 2nd-best to 3rd-best etc)	
(higher values in 1st best than 10th best)	**(lower in 1st best)**
sblm-nts delta=-0.003010888 N=4058	sblm-pword delta=0.009007925 N=4058
spc[NPB][DT] delta=-0.0010995 N=4003	spc[SG][VP] delta=0.0275367 N=1863
sb[<s>][DT] delta=-0.00106046 N=4003	sb[<s>][NN] delta=0.01770083 N=2984
sblm-unkword delta=-5.227235e-05 N=4058	spc[NP-C][,] delta=0.04496575 N=1185
spc[MD][MD] delta=-0.02748918 N=6	spc[NP][NPB] delta=0.02352679 N=2397
sb['']['NNS] delta=-0.02900433 N=5	sb[NP-C][,] delta=0.04613019 N=1342
spc[SBARQ][SBAR] delta=-0.06666667 N=2	spc[S][S] delta=0.04400232 N=1424
sb[NN][IN] delta=-0.02200577 N=6	spc[VP][PP] delta=0.02611785 N=2411
spc[SBARQ][NP] delta=-0.06212121 N=2	sb[,][VP] delta=0.06431472 N=1015
	spc[VP][VP] delta=0.04374572 N=1511
	sblm delta=0.01814439 N=4058

Excuses

The models (PCFG and category-pair indicators) are not good. Lexicalize or include grandparent context. Qing showed better perplexity using the aunts and grandmothers of the RHS nonterminals.

We already indirectly optimize the models (you can see that the system 1-best has better sblm probability than the 10th-best.

LM training data is just bitext (I still intend to try same- style parses of large monolingual data but don't expect much improvement).

MT rules don't really offer much CFG-rewrite variation across restructured-tree-nodes when restricted to a particular foreign sentence (this is consistent with my not needing to decode greedily).

Variations

Open vs closed (context-dependent) p(<unk>). Memorize -0 and -1 split rewrites for PCFG. Don't skip @-xxx-BAR nodes. Greedy vs not. Just sibling sb indicators. Just sblm. Just parent-child spc indicators. My python LM training vs SRI LM training (nearly identical perplexity, so expect no difference)