Creating a rapid falloff in the proportion of matchings permitted (Wu 1995a). This characteristic gives ITGs inherently permit nearly free matchings for fanouts up to four, with strong constraints thereafter stemming from the compositionality principle, this constraint is important for computational reasons, to avoid the subtrees' immediate parent constituents are also matched to each other. Aside from linguistic motivations, crossing constraints are otherwise impossible to match the singletons. The result is that the maximum-likelihood parser selects the translation lexicon with degrees of probability on each potential word translation. A small frames of reference may each be more adequately discriminating to match constituents, without language-specific monolingual grammars for the source and target languages, simply by bringing the ITG constraints to bear in tandem with lexical matching. Thus, a parser with this grammar can build a bilingual parse tree for any possible ITG matching translations. The form is called BITG (bracketing ITG).
biparsing

aim
- automatically parse an input/output sequence pair

Input
- bisequence (input-output pair of sequences)
- token translation lexicon
- language-independent bracketing ITG (only 1 generic nonterminal: A)

output
- aligned bisequence

Implementation
- biparser

2-normal form for ITGs

Lecture 3
For any inversion transduction grammar G', there exists an equivalent inversion transduction grammar G of the form (S1, T1, T2), such that the left-hand side of any production in G' contains a single terminal pair or a list of terminal pairs.

Theorem 1
For any inversion transduction grammar G, there exists an equivalent inversion transduction grammar G' in which every production takes one of the following forms:

1. A → x/y, A → x/c, A → y/c, A → [B2X2]
2. A → a
3. A → c/c, A → x/G, A → y/G, A → [BIB2]

(A, c/c)
translation-driven segmentation

aim
- avoid premature segmentation errors
- optimize chunk boundaries using bilingual/bimodal clues

input
- bisequence (input-output pair of sequences)
- token translation lexicon

output
- segmented bisequence

implementation
- segmental biparser, instead of token-based biparser
- integrates segmentation decisions into dynamic programming

[Chen, 2005]

coercion

aim
- bootstrap grammar induction, knowing only some other language
- use bilingual/bimodal clues to coerce Chinese into English tree structure

input
- bisequence (input-output pair of sequences)
- token translation lexicon
- language-independent bracketing ITG (only 1 generic nonterminal A)

output
- labeled tree structure for output language sequence

implementation
- parser (choose either token-based or segmental version)

[Wu, Y. 1995]
Learning phrasal translation lexicons

<table>
<thead>
<tr>
<th>% in real</th>
<th>% in model</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>5%</td>
</tr>
</tbody>
</table>

Would you take an acceptable starting point for this new policy here and now, as about 3.5 million people have already done so in Hong Kong?

In what way the Government would increase their job opportunities, and the teachers and students to whom they would give the principles, and[1] last month was to say "never" or "never be used", reserving and expiring the consent or expiring the consent, and therefore in the practical difficulties in the sentence.

An effective representation is elegant

Turing machines are too inefficient to learn

Transduction ("decoding")

**aim**

- translate a sequence from an input representation to an output representation, with zero syntactic knowledge

**Input**

- input language sequence
- token translation lexicon
- language-independent bracketing ITG (only 1 generic nonterminal A)

**Output**

- output language sequence that is a translation of the input sequence

**Implementation**

- CKY style BITG transducer with output language n-gram model

An effective language is elegant
transduction ("decoding")

**aim**
translate a sequence from an input representation to an output representation, with zero syntactic knowledge

**input**
- input language sequence
- token translation lexicon
- language-independent bracketing ITG (only 1 generic nonterminal A)

**output**
- output language sequence that is a translation of the input sequence

**implementation**
- CKY style BITG transducer with output language n-gram model

---

transduction grammar induction

**aim**
go from a parallel corpus to a transduction grammar which can then be used as a transducer (translation system)

**input**
- bicornus (large set of bisequences)

**output**
- transduction grammar (typically an ITG)

**implementation**
- incrementally learn TGs from simple to complex
- gradually climb up the complexity hierarchy for translations

---

EM algorithm for ITGs

**input**
- input language sequence
- token translation lexicon
- output language CFG, with each rule mirrored into straight & inverted ITG rules

**output**
- output language sequence that is a translation of the input sequence

**implementation**
- Earley style ITG transducer with output language n-gram model

---

we've barely scratched the surface of the many applications of stochastic transduction grammars

- grammar induction: automatically decomposing both sides of a parallel corpus
- formalizing
- inferring: bilingual parsing of both sides of a parallel corpus
- inducing: induct an ambient context to extract or induce transduction rules (grammatical, transfer rules, examples, templates)
- segmenting: translation-driven word segmentation ("syran-kaw")
- coercion or projection: parsing one language using the grammar of another language
- grammatical lattice learning & decoding: particularly useful for linguistic grammars (which are often not linear)
- training: polynomial-time EM parameter estimation
- translations or decoding: polynomial-time statistical translation
- grammatical decoding: even faster, grammatical statistical machine translation (incorporating linguistic grammar)
- transduction grammar induction: fully unsupervised learning of relations

Efficient polynomial-time algorithms for PFGT, ITG, and ITG learning of structural relationships are possible because of the evolutionary drive toward restricted expressiveness of practical languages and concepts in common use.

---

EM algorithm for ITGs

**input**
- input language sequence
- token translation lexicon
- output language CFG, with each rule mirrored into straight & inverted ITG rules

**output**
- output language sequence that is a translation of the input sequence

**implementation**
- Earley style ITG transducer with output language n-gram model

---

but a lot of SMT today

- relies excessively on
  - memorizing long lexical translations
  - ... or heavily lexicalized long translation rules
- rather than
  - generalizing abstract patterns
  - reusing short lexical translations
wait...

you wanna translate with finite-state transducers?!

- no
  - FSTs are useless for anything but the most simple, monotone translation tasks
- but...
  - complex translation tasks are made up of simple, monotone translation tasks

review

a grammar generates a language
... which is a set of sentences

a transduction grammar generates a transduction
... which is a set of sentence pairs
**Linear**

- Linear grammar
  - $S \rightarrow A$
  - $A \rightarrow \varepsilon$
  - $A \rightarrow eB$
  - $A \rightarrow Be$

- Linear transduction grammar
  - $S \rightarrow A$
  - $A \rightarrow \varepsilon$
  - $A \rightarrow eB$
  - $A \rightarrow e/fB$

**Lots of rule forms...**

- ... for the same token pairs (biterminals)
- Risks diluting the statistics
- Introduce preterminalized transduction grammars

**PLIC parsing**

- PLITG parsing
**PLITG parsing**

```
PLITG parsing
PLITG parsing
PLITG parsing
```

**ITG**

- context-free grammar (2-normal form)
  - $S \rightarrow A$
  - $A \rightarrow BC$
  - $A \rightarrow e$

- inversion transduction grammar
  - $S \rightarrow A$
  - $A \rightarrow [BC]$
  - $A \rightarrow \langle BC \rangle$
  - $A \rightarrow e/f$
  - $A \rightarrow e/\epsilon$
  - $A \rightarrow \epsilon/f$

**ITG parsing**

```
PLITG parsing
PLITG parsing
PLITG parsing
```

**Roadmap**

- Grammar conversion: PFSTG-PLITG
  - PFSTG
    - $S \rightarrow A$
    - $A \rightarrow \epsilon/\epsilon$
    - $A \rightarrow PB$
  - P
    - $P \rightarrow e/f$

- Grammar conversion: PLITG-ITG
  - PLITG
    - $S \rightarrow A$
    - $A \rightarrow \epsilon/\epsilon$
  - ITG
    - $S \rightarrow A$
    - $A \rightarrow \epsilon/\epsilon$

- Grammar conversion: PFSTG-ITG
  - Cheat!
  - Compose the previous conversions PFSTG-PLITG • PLITG-ITG gives PFSTG-ITG
  - No training at the PLITG stage

- Not as straight forward
  - Idea: "promote" preterminals to be proper nonterminals
    - $P \rightarrow e/f$ becomes $P \rightarrow e/f$ and $P \rightarrow A$
    - $P(e/f|P) = \alpha p(e/f|P)$,
      $p(A|P) = (1 - \alpha) \beta(A|P) p(e/f|P)$
    - $\alpha = 0.5$, $\beta(A|P) = \text{uniform over nonterminals}$
  - Perform standard grammar normalization to eliminate nullary and unary rules
Fun with simple grammars

- Splitting
  - Split one nonterminal or preterminal symbol into n new symbols
  - Apply controlled perturbation to split the probability mass
  - (See the paper for details)

Splitting helps!

- Assume two rules:
  - \( A \to [P A] \)
  - \( P \to \text{will/要} \)
- We want to:
  - Split \( A \) into \( A \) and \( B \)
  - Split \( P \) into \( P \) and \( Q \)
- Resulting rules:
  - \( A \to [P A], A \to [Q A], A \to [Q B] \)
  - \( B \to [P A], B \to [P B], B \to [Q A], B \to [Q B] \)
  - \( P \to \text{will/要}, Q \to \text{will/要} \)

Fun with simple grammars

- Chunking
  - Any sequence of two contiguous terminal productions found in the parse forest could be one production
  - Each round of chunking doubles the potential phrase length
  - Time consuming on large forests
  - (See our EAMT and Interspeech paper from last year for details)

Chunking helps!

- how long will it take on foot?
**experimental setup**
- Initialize a PFSTG from a parallel corpus
  - IWSLT07 Chinese-English
- Chunk and split to your heart's desire
- Move to PLITG then to ITG or move straight to ITG
- Traverse the roadmap
- Train at each stage
- Measure cross-entropy

**why is cross-entropy on the training set important?**
- Low cross-entropy may be indicative of over-fitting
- Over-fitting to a lower cross-entropy is indicative of model fit
- PFSTGs are unable to over-fit to the same level that ITGs are able to
- Indicates that ITGs as a model is a better fit to the problem

**lessons ways to lower the entropy**
- Baseline: cross-entropy 110.2 (PFSTG)
- Best: cross-entropy 32.8 PFSTG-chunk-chunk-split-split-split-ITG
  - Nice: still fits in 16Gb RAM
- Promising candidate: cross-entropy 35.9 PFSTG-chunk-chunk-ITG
  - After only 1 chunk: 60.7
  - After chunk-chunk: 36.5 (already mostly there!)
  - Unfortunately, couldn't split after chunk-chunk-chunk within 16Gb RAM

**remember what we want**
- big parallel corpus → small transduction grammar
- learn lexical phrase translations (i.e., a segmental transduction grammar)
- compact generalization of the translation knowledge encoded in the corpus
- unsupervised learning of transduction grammar rules without Giza, Moses, parsers, or anything else

**why?**
- rearchitecting the SMT core: "Machine Learning 101"
  - Do training and testing on the same model
  - Get the inductive bias right: core internal representation designed from the start for learning semantic frame generalizations
  - Emphasis on generalizing rather than memorizing
  - Minimum description length / MAP → Occam's razor for model size
- evaluated in pure, unadulterated form
  - Not as a preprocessing subroutine (eg, for word alignment) within an off-the-shelf "stack-of-hacks" SMT spaghetti architecture
  - ITG decoder matched to ITG learner
  - We'd rather see lower BLEU scores temporarily, so we can better understand transduction grammar induction behavior
  - Don't obscure your model by burying it within a big "stack-of-hacks!"
common SMT training pipeline
- Long pipeline propagates errors: risk of premature commitment
  - No way to recover from mistakes in earlier steps
  - No credit/blame assignment during learning!
  - To try to compensate: massively over-generate phrases
    - Result: Heavy bias toward memorizing huge corpus instead of learning the right abstract generalizations
  - Same pipeline problem for Hierarchical/Syntactic (tree, string) -> (tree, string)
  - Large parallel corpus where the trees go
  - Replace "phrases" with TG rules

Our training "pipeline"
- No pipeline
- No risk of premature commitment
- Replaces many intermediate learning steps
  - ...which, admittedly, have been engineered with heuristic tweaks over a long time
  - Worth it to carefully understand correct unsupervised learning of generalizations

Our specific SSBITG model
  - Unsupervised induction of stochastic segmental bracketing inversion transduction grammars
    - Compact generalization of the translation knowledge encoded in the corpus
  - Bayesian learning objective
    - Closely related to minimum description length
    - Structural prior based on minimum description length
    - Dirichlet parameter prior
    - Fixed model type prior
  - Structural search based on iteratively segmenting sentence pairs

How to induce transduction rules?
- Can we beat our COLING 2012 bottom-up chunking method?
- Test by using induced transduction grammar directly to translate unseen sentences

Bottom-up rule chunking strategy
1. Initialize a token-based FSTG
2. Parse the bicorpus
3. Assume that any two adjacent lexical productions could have been generated with a single chunked rule, and add them
4. Train the FSTG using EM
5. Transform the FSTG into an LIITG and train using EM
6. Transform the LIITG into an ITG and train using EM

Saers, Markus, Karteek Addanki & Dekai Wu (2012)
"From Finite-State to Inversion Transductions: Toward Unsupervised Bilingual Grammar Induction" in COLING 2012
**Opposite search strategy “directions” for learning transduction rules**

- **Chunk rules bottom-up** (COLING 2012)
  - Start with very small lexical equivalences (biterms)
  - Look for promising chunks during biparsing
  - Make the chunks explicit biterms
  - Repeat
- **Segment rules top-down** (IJCNLP 2013)
  - Start with all sentence pairs as biterms
  - Segment the existing biterms into smaller chunks
  - Make the smaller chunks explicit biterms
  - Repeat

---

**Segmentation, intuitively**

- **five thousand yen is my limit**
  - 我最多出五千日元
- **the total fare is five thousand yen**
  - 总共的费用是五千日元

---

**New search strategy**

**Top-down rule-segmenting strategy**

1. initialize an ITG containing each sentence pair as a biterm
2. foreach "frequently shared biaffix"
3. hypothesize the set of segmentations that it suggests
4. evaluate the delta in ITG “goodness” the set would cause*
5. commit greedily to good hypothesis sets
6. goto 2

* for present purposes, “goodness” = description length

---

**“Goodness” of an ITG**

If objective function is description length...

- #bits needed to encode the model DL(Φ)
  - Φ = model = ITG
- plus...
  - #bits needed for the model to encode the data DL(D|Φ)
  - D = data = parallel training corpus

---

**Iterative rule segmentation**

as phrasal translation lexicon search

**Q.** How to suggest possible ways to segment rules?

**A.** Look for frequently shared biaffixes

- Each biaffix suggests a set of rule segmentation hypotheses
  - 4 types of rule segmentations can be suggested
  - Can be efficiently computed
  - Estimate delta in objective function for each
Top-down rule segmentation

S → A
A → < AA >
A → five thousand yen/五千日元
A → is my limit/我最多出

A → [ AA ]
A → the total fare is/总共的费用是
A → five thousand yen/五千日元

Description length of an ITG

- Serialize grammar into a message
- Measure number of bits needed to encode the message
- Example:
  \[ S \rightarrow A \quad A \rightarrow \langle AA \rangle \quad A \rightarrow [AA] \]
  \[ A \rightarrow \text{have/有} \quad A \rightarrow \text{yes/是} \]
- Becomes the message:
  \[ < AA > \quad \text{have/有} \quad \text{yes/是} \]

- Assume each symbol requires \( \log \frac{1}{N} \) bits (where \( N \) is the total number of symbols)
- The above message contains 8 unique symbols → 3 bits each
- The message is 23 symbols long, and needs \( 23 \cdot 3 = 69 \) bits to encode

Minimum Description Length objective

- Want: \( \text{argmin} DL(\Phi, D) \)
  \[ DL(\Phi, D) \propto DL(D | \Phi) + DL(\Phi) \]
  \[ DL(D | \Phi) = -\log P(D | \Phi) \]

MAP vs. MDL

- Maximum a posteriori probability
- Minimum description length
- Relationship:
  \[ DL(x) = -\log P(x) \]
  \[ P(x) = 2^{-DL(x)} \]
- Enables:
  - Probabilistic formulation of description length search
  - Description length formulation of probabilistic search

Bayesian search (MAP)

- Maximize a posteriori probability of the model given the parallel corpus:
  \[ P(\Phi | D) \propto P(\Phi) P(D | \Phi) \]
- Decompose the model prior such that:
  \[ P(\Phi) = P(\Phi_C) P(\Phi_S | \Phi_C) P(\theta_B | \Phi_S, \Phi_C) \]
  \[ P(\Phi_C) = \text{bracketing inversion transduction grammar} \]
  \[ P(\Phi_S | \Phi_C) = 2^{-DL(\Phi_S | \Phi_C)} \]
  \[ P(\theta_B | \Phi_S, \Phi_C) = \text{symmetric Dirichlet distribution} (\alpha = 2) \]

Bayesian search (MAP)

- Full search problem:
  \[ \text{argmax} P(\Phi_C) \times P(\Phi_S | \Phi_C) \times P(D | \Phi_S, \Phi_C, \theta_B) \]

In MDL:

- \[ \text{argmin} DL(\Phi_C) + DL(\Phi_S | \Phi_C) + DL(D | \theta_B, \Phi_S, \Phi_C) \]

Evaluating the delta in \( P(D | \Phi) \)

- Requires biparsing for every hypothesized new \( \Phi \)
- Intractable

... must approximate

estimating the delta in \( P(D | \Phi) \)

- Assume that \( r_0 \) is segmented into \( r_1, r_2 \) and \( r_3 \)
- Approximation assumption:
  \[ DL(D | \Phi_C, \Phi_S, \theta_B) = DL(r_1) + DL(r_2) + DL(r_3) \]

- The new rule probability function \( \hat{P} \) will be:
  \[ \hat{P}(r_1) = \beta(n) + \frac{\hat{P}(r_1)}{\hat{P}(r_3)} \]
  \[ \hat{P}(r_2) = \beta(n) + \frac{\hat{P}(r_2)}{\hat{P}(r_3)} \]
  \[ \hat{P}(r_3) = \beta(n) + \frac{\hat{P}(r_3)}{\hat{P}(r_3)} \]

transduction grammar induction by top-down segmenting transduction rules

\[ G \]
\[ \text{// The grammar} \]
\[ \text{biaffixes_to_rules} \]
\[ \text{// Maps biaffixes to the rules they occur in} \]
\[ \text{biaffixes_delta} = \[] \]
\[ \text{// Hypothesized biaffixes' impact on P(D|G)} \]

for each biaffix \( b \):

- delta = eval_dlt(b, biaffixes_to_rules(b), G)
- if (delta < 0)
  - biaffixes_delta.push(b, delta)
- sort_by_delta(biaffixes_delta)

for each b:delta pair in biaffixes_delta:

- real_delta = eval_dlt(b, biaffixes_to_rules(b), G)
- if (real_delta < 0)
  - G = make_segmentations(b, biaffixes_to_rules(b), G)
impact of model structure
a posteriori probability during learning (log domain)

Impact of model structure
rule count during learning (log domain)

BLEU score

NIST score

(in table form)

<table>
<thead>
<tr>
<th>System</th>
<th>NIST</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>0.8554</td>
<td>8.83</td>
</tr>
<tr>
<td>initial</td>
<td>0.0000</td>
<td>0.00</td>
</tr>
<tr>
<td>iteration 1</td>
<td>0.6686</td>
<td>9.38</td>
</tr>
<tr>
<td>iteration 2</td>
<td>3.9976</td>
<td>15.30</td>
</tr>
<tr>
<td>iteration 3</td>
<td>4.3928</td>
<td>17.89</td>
</tr>
<tr>
<td>iteration 4</td>
<td>4.3122</td>
<td>16.26</td>
</tr>
<tr>
<td>iteration 5</td>
<td>4.0981</td>
<td>16.10</td>
</tr>
<tr>
<td>iteration 6</td>
<td>3.9191</td>
<td>15.97</td>
</tr>
<tr>
<td>iteration 7</td>
<td>3.8338</td>
<td>15.06</td>
</tr>
</tbody>
</table>

MDL/MAP transduction grammar induction

- Bayesian MAP quite promising for driving ITG induction via top-down segmentation of rules
- closely related to MDL (DL is more natural for ITG structure prior)
- Beats bilingual lexical chunking driven by ML
  - learns a much smaller ITG...
- ...that performs better on held-out test data
- New! even better results combining chunking and segmentation
- Reconstructing the SMT core: “Machine Learning 101”
  - inductive bias – internal representation is set up from the start for learning semantic frame generalizations
  - learns small models – less reliance on memorizing huge corpora
  - rapidly improving – newer results already into mid-20s BLEU range without Giza, Moses, parsers, or anything else
  - representational transparency – error analyses to understand learning properties

Can the same exact core capability be used not only for conventional AI tasks, but also for all sorts of creative tasks?

escaping from blocks worlds

transduction grammars simultaneously model
Boden’s (1992) three types of creativity

- **combinational** new combinations of familiar ideas
- **exploratory** generation of new ideas by exploration of a space of concepts
- **transformational** involves a transformation of the search space so new kinds of ideas can be generated

what makes music?
the many languages of music

lyrics
sequences of words
melodies
sequences of notes
chord progressions
sequences of chords
rhythms
sequences of pulse hits
meters
sequences of repetitions
ostinatos
sequences of lines
verses
sequences of stanzas
songs
sequences of verses
dynamics
sequences of volumes

it's about the relationships

• the relationships between multiple different musical languages differentiate aesthetically pleasing music from jarring noise
• internalizing these relationships accounts for
  • expectation
  • surprise
  • resolution

Why apply NLP to music?

• Music
  • is a form of language
  • has had major impact across all human cultures
  • emerges from similar cognitive processes as speech and written language (McMullen and Saffran 2004)
• Applying NLP to music
  • similar generalizations to be captured in music and natural language
  • distinguishes well-motivated learning methods from language specific flaws
  • adaptation of statistical NLP models presents interesting challenges

Hip hop is spoken language.

• rap is one of the world’s most popular forms of spoken language (for decades!) – arguably spoken language’s most significant development in ages yet inexplicably ignored in language research
• musical lyrics way more challenging than classical poetry due to absence of traditional constraints
  • far fewer meter restrictions
  • variable rhyme schemes
  • unusual vocabulary (Bill Gates ≈ 40 ill dates)
• user generated content available online
• but off-the-shelf NLP tools not suitably designed

freestyling

• freestyle: improvisational style of rap performed “off the top of the head” with no previously composed lyrics
• freestyle battle: contest of rappers dueling by challenging and responding using improvised lyrics

Steve Jobs vs. Bill Gates

Jobs
A man uses the machines you built to sit down and pay his taxes

Gates
Well, Steve, you steal all the credit for work that other people do. Did your fat beard Wozniak write these raps for you too?
verse vs. stanza vs. line

- All right stop collaborate and listen
- Ice is back with my brand new invention
- Something grabs a hold of me tightly
- Flow like a harpoon daily and nightly
- Will it ever stop ya, I don’t know
- Turn off the lights, and I’ll glow
- To the extreme I rock a mic like a vandal
- Light up a stage and wax a chump like a candle
- Dance go rush to the speaker that booms
- I’m killing your brain like a poisonous mushroom
- Deadly when I play a dope melody
- Anything less than the best is a felony
- Love it or leave it you better gain weight
- You better hit bull’s eye the kid don’t play
- If there was a problem, yo I’ll solve it
- Check out the hook while my DJ revolves it

(REESTYLE)

FREESTYLE model

- approach – learn to “translate” any line of hip hop to produce improvised, rhyming lyrics
- requirements – the underlying model must have
  - strong enough inductive bias for learning to generate responses even without any linguistic or phonetic knowledge
  - sufficient expressive capacity to represent structural relationships between lyrical lines
- model – bracketing inversion transduction grammar (BITG) induction
  - rule learning via chunking vs segmentation
  - data selection via adjacent lines vs rhyme scheme detection
  - for English vs Maghrebi French hip hop

transducing challenges to responses

- HKUST ITG decoder
  - bottom-up CKY style parsing algorithm with cube pruning (Chiang 2008)
- monotonic constraint
- bias to match the rhyming order of challenge
transducing challenges to responses

- HKUST ITG decoder
  - Earley style transducer supports arbitrary non-normal-form, non-binarized ITGs of any rank with mixed terminals/nonterminals in rules [Yu & Wong 1998]
  - segmental ITG support any length lexical phrase translations (Yu & Wong 1997)
  - rules pruning (Chiang 2008)
- penalize reflexive rules like A = yeah / yeah
- challenge and response identical in ~5% of training instances (chorus lines)
- danger: reflexive rules that map a line back to itself get induced
- so decoder penalizes responses too similar to the challenges
- also decoder can penalize inverted rules as a bias to match rhyming order

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hello hello hello hello</td>
<td>Hello hello hello hello</td>
</tr>
<tr>
<td>yeah yeah yeah yeah</td>
<td>yeah yeah yeah yeah</td>
</tr>
<tr>
<td>ain't ain't ain't ain't</td>
<td>ain't ain't ain't ain't</td>
</tr>
<tr>
<td>can't can't can't can't</td>
<td>can't can't can't can't</td>
</tr>
</tbody>
</table>

---

issues for evaluation

- systems to be judged on quality of responses
- but what is "quality of responses"?
  - absence of improvised "references"
  - extremely subjective akin to "translation"
- no automatic metrics like BLEU exist
- multiple evaluation criterion
- larger context necessary for evaluation

---

disfluency in hip hop lyrics

- ~10% of data had successive repetitions of words like the and I
- disfluencies typically result from repetitive chants, exclamations, and interjections in lyrics
  - phrase style lyrics
    - "hypermam" style backing vocals
  - compare two disfluency handling strategies
    - filtering – remove lines with disfluencies
    - correction – replace all successive repetitions (the the the --- the)

---

data selection?

- data
  - 2,000 lyrics, 260,000 verses
  - 4,2M tokens and 153,000 token types
- Small fraction of Arabic, French and Spanish lyrics
- phrase based SMT baseline (PBSMT)
- evaluate out-of-the-box SMT performance
- Small fraction of Arabic, French and Spanish lyrics
- 4.2M tokens and 153,000 token types
- 52,000 lyrics, 260,000 verses
- 4-gram LM trained on all the lyrics

---

creating training data

- how do we select challenge-response pairs?
  - need – lots of training examples consisting of
    - a line of rap, with
    - a fluent and salient rhyming response
  - naïve approach – all pairs of lines in the same stanza
    - explodes the training data size
  - keeps training set size proportional to rap corpus size
  - does not capture rhyming dependencies
  - better approach 1 – all successive line pairs in the same stanza
    - keeps training set size proportional to rap corpus size
    - but still does not ensure that training examples rhyme
  - better approach 2 ("RS") – only successive line pairs that rhyme
    - but how can we know which line pairs actually rhyme?
rhyme scheme detection
(Addanki & Wu, SLSP 2013)

- in keeping with our linguistics-lite model – we don’t use a pronunciation dictionary
- hip hop rhyming often defies mainstream pronunciations
- want a language-independent model
- instead – identify rhyming lines using a rhyme scheme detector

result — data selection
disfluency correction + rhyme scheme detection

- disfluency correction is better than disfluency filtering
  - improves both fluency and (surprisingly?) rhyming
- rhyme scheme detection
  - improves fluency for both TG and PBSMT models
  - improves the fraction of sentences with ≥ acceptable rhyming
  - similar results can be observed for ISTG models also (coming up)

versus

verse vs. stanza vs. line

line 1
All right stop collaborate and listen
'Tis back with my brand new invention
Flow like a harpoon daily and nightly
Will it ever stop, I don’t know?

line 2
To the extreme I rock a mic like a candle
Light up a stage and wax a chump like a candle
Dance go rush to the speaker that booms
I’m killing your brain like a poisoned mushroom

line 3
Deadly when I play a dope melody
Anything less than the best is a felony
Love it or leave it you better gain weight
You better hit bull’s eye the kid don’t play

line 4
If there was a problem, yo’ll solve it
Check out the hook while my DJ resolves it

verse


rhyme scheme detection
(Addanki & Wu, SLSP 2013)

generative model for verses

- generated by a fully connected HMM
  - each state $S_i$ is a stanza with a particular rhyme scheme $r$
  - emissions are a sequence of final tokens $x_{t,n}$ in each line of the stanza
  - a single textual line of lyrics might contain two lyrical lines separated by a comma

transduction grammar induction

- bracketing inversion transcription grammars (BITG)
  - empirically high coverage, accuracy across various NLP tasks
  - sufficient expressiveness to handle word associations between lines
  - efficient induction and decoding algorithms

- compare two approaches, trained on same amount of data
- TG token-based BITG
- ISTG interpolated segmental BITG
- token-based BITG captures word associations better
  - efficient induction algorithm using DPH
  - beam pruning
  - MDL driven induction
  - segmental BITG captures phraseal associations better
  - efficient induction algorithm using MDL induction

chuncking vs segmentation?

transduction grammar induction
token-based vs. segmental grammars

- **transduction grammar rules can contain**
  - rules \( A \rightarrow \{B, A, f, A, l, f, l, w, o, n, g, v, w, r, o, n, g\} \)
  - lexical rules \( A \rightarrow \{f, l, l, b, a, d\} \)
  - structural rules \( A \rightarrow \{A, A\} \)
- **segmental transduction grammars**
  - biterminals can contain multiple tokens in each language
  - simple and efficient learning algorithms
- **token-based** transduction grammars
  - biterminals contain at most one token in each language
  - unsupervised learning algorithms
  - suffer from a lack of fluency in the output

interpolated segmental model

- **segmental ITG learned using a top-down rule**
  - segmentation approach driven by a MAP/minimum description length objective
  - aim: balance fluency and expressivity by interpolating the token-based ITG \( G_a \) with a segmental ITG \( G_b \)
- **segmental ITG learned using a top-down rule**
  - segmentation approach driven by a MAP/minimum description length objective
  - aim: balance fluency and expressivity by interpolating the token-based ITG \( G_a \) with a segmental ITG \( G_b \)
- **learning segmental bigrammars**
  - most SMT approaches instead
  - use heuristics to extract segments from token alignments
  - typ. justified by BLEU motivations not applicable to our task

result — English hip hop

challenge-response test examples

<table>
<thead>
<tr>
<th>Challenge</th>
<th>man i die to see em all thun i just don't care</th>
</tr>
</thead>
<tbody>
<tr>
<td>FREESTYLE</td>
<td>in the sky and me in pillon and the you there</td>
</tr>
<tr>
<td>PBSMT</td>
<td>man i like to see em all i just don't care</td>
</tr>
<tr>
<td>Challenge</td>
<td>did a twelve year bid in the streets and hold it down</td>
</tr>
<tr>
<td>FREESTYLE</td>
<td>to the girls here kid the and to the thought the now</td>
</tr>
<tr>
<td>PBSMT</td>
<td>did a year in the streets and it down</td>
</tr>
<tr>
<td>Challenge</td>
<td>oh i believe in yesterday</td>
</tr>
<tr>
<td>FREESTYLE</td>
<td>can you see the day</td>
</tr>
<tr>
<td>PBSMT</td>
<td>oh i believe in tomorrow</td>
</tr>
<tr>
<td>Challenge</td>
<td>what would i do</td>
</tr>
<tr>
<td>FREESTYLE</td>
<td>just me and you</td>
</tr>
<tr>
<td>PBSMT</td>
<td>what would you do</td>
</tr>
<tr>
<td>Challenge</td>
<td>cause you aren't going home till the early morn</td>
</tr>
<tr>
<td>FREESTYLE</td>
<td>and the you this alone i gotta on</td>
</tr>
<tr>
<td>PBSMT</td>
<td>cause you and your friends ain't nothing but</td>
</tr>
</tbody>
</table>
**result — learned transduction grammar rule examples**

<table>
<thead>
<tr>
<th>transduction grammar rule</th>
<th>log prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A \rightarrow \text{long/wrong})</td>
<td>-11.6747</td>
</tr>
<tr>
<td>(A \rightarrow \text{rhyme/time})</td>
<td>-11.6694</td>
</tr>
<tr>
<td>(A \rightarrow \text{felt bad/couldn't see what I really had})</td>
<td>-11.3196</td>
</tr>
<tr>
<td>(A \rightarrow \text{matter what you say/leaving anyway})</td>
<td>-11.8792</td>
</tr>
<tr>
<td>(A \rightarrow \text{arhythmatism/this rhythm is sick})</td>
<td>-12.3492</td>
</tr>
</tbody>
</table>

**result — token-based vs segmental**

Segmental TG improves fluency and rhyming

<table>
<thead>
<tr>
<th>Segment</th>
<th>Fluency</th>
<th>Rhyming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Token</td>
<td>57.46%</td>
<td>70.64%</td>
</tr>
<tr>
<td>Segment</td>
<td>59.44%</td>
<td>71.88%</td>
</tr>
<tr>
<td>1.5%</td>
<td>59.44%</td>
<td>71.88%</td>
</tr>
<tr>
<td>0.5%</td>
<td>59.44%</td>
<td>71.88%</td>
</tr>
<tr>
<td>0%</td>
<td>59.44%</td>
<td>71.88%</td>
</tr>
</tbody>
</table>

- Interpolated segmenting TG (ISTG) produces more fluent responses than token-based TG on both data sets (more later)
- ISTG also produces better rhyming responses (surprising?)
- Results also demonstrate that off-the-shelf phrase-based SMT systems (PBSMT) cannot be directly adopted for this task

**Maghrebi French hip hop**

**learning to freestyle in Maghrebi French**

- Advantage of our linguistics-lite model is that it should work independent of language
- Test if it can learn to generate response lyrics in languages other than English
- Initial experiments on Maghrebi French hip hop lyrics
- Our model performs surprisingly well despite
  - No special adaptation
  - Much smaller training data size
  - Diverse and noisy training data

**Maghrebi French hip hop corpus**

- About 1300 hip hop song lyrics
- Majority in Maghrebi French: French interspersed with romanized Arabic, Arabic and English phrases
  - De la traversée du désert au bon coussus de Yéma (Yéma = My mother)
  - Le yommi mia = my son / a thick with = A bee
  - Tes game over, game over / Le son de Chicken wings
- Linguistically complex — language dependent models will be hard to adapt
- 47,000 sentence pairs selected using rhyme scheme detection

**result — Maghrebi French hip hop**

**learned transduction grammar rule examples**

<table>
<thead>
<tr>
<th>transduction grammar rule</th>
<th>log prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A \rightarrow \text{terre/la guerre})</td>
<td>-9.4837</td>
</tr>
<tr>
<td>(A \rightarrow \text{haine/peine})</td>
<td>-9.7706</td>
</tr>
<tr>
<td>(A \rightarrow \text{mal/pays natal})</td>
<td>-10.6877</td>
</tr>
<tr>
<td>(A \rightarrow \text{le frisson/mi corazon})</td>
<td>-11.0931</td>
</tr>
<tr>
<td>(A \rightarrow \text{gratteurs/rappeurs})</td>
<td>-11.7306</td>
</tr>
</tbody>
</table>

**result — Maghrebi French hip hop**

**conclusions**

- **FREESTYLE**: first known model for learning how to rap battle
  - Translates challenge lyrics to improved responses
  - Transduction grammar induction is fully unsupervised
  - learns fluent, rhyming responses absent linguistic knowledge
  - New MDL driven learning of hip hop ITG rules by segmentation
  - Segmental grammars improve on token-based grammars
  - Both outperform off-the-shelf PBSMT contrastive baseline
  - Hip hop domain specific models against very noisy training data yield more fluent and better rhyming responses
  - Data selection via unsupervised rhyme scheme detection model
  - Difficulty connection
  - Completely unsupervised generation of hip hop challenge responses without linguistic knowledge despite the noisy domain
  - Generalizes to non-English hip hop: encouraging results on Maghrebi French validate language independence assumptions
bilingual vector space models

- dearth of vector models for compositional learning of bilingual relations
- predominantly augment "shake-n-bake" SMT modeling assumptions using feature vectors
- n-gram translation models (Soricut et al., 2012)
  - bilingual generalization of class-based n-grams using distributed representations
  - fails to model compositionality and cross-lingual reordering
- bilingual word embeddings (Zhou et al., 2015)
  - recurrent INLMs with SMT word alignments
  - only learns non-compositional features

bilingual vector space models

- INLMs = input language context (Devlin et al., 2014)
  - does not model input and output language features simultaneously
- recurrent probabilistic models (Kochmann & Blunsom, 2011)
  - generates an input sentence representation that generates an output sentence
  - lacks structural constraints and relies on a LTM to reorder output
- reordering prediction using RAEs (Li et al., 2013)
  - monolingual RAEs to predict reordering in a monolingual ITG model
  - uses only input language context

maximum a posteriori objective

\[
\arg \max_{\phi} P(\phi) P(\mathcal{D} | \phi)
\]

bilingual corpus likelihood

\[
P(\phi) = P(\phi_2 | \phi_1) P(\phi_1) = \prod_{n=1}^{N} P(\phi_n | \phi_{n-1})
\]

ML prior

\[
-\log P(\phi_1, \phi_2) = \frac{1}{\alpha} \sum_{n=1}^{N} \phi_n
\]

Dirichlet prior

\[
P(\phi_1, \phi_2) = \left( \sum_{n=1}^{N} \phi_n \right)^{-\alpha} \prod_{n=1}^{N} \phi_n^{\alpha-1}
\]

putting everything together, we want

\[
\arg \max_{\phi} P(\phi_1, \phi_2 | \phi_2 = \phi_1) P(\phi_1 | \phi_2 = \phi_1) P(\mathcal{D} | \phi_1, \phi_2, \phi_2 = \phi_1)
\]

challenges with symbolic ITGs

- unsupervised ITG induction remains hard
  - extremely large model space!
- nonterminals don’t capture context efficiently
  - intractable – each context has explicit nonterminal
    - solution: replace nonterminals with feature vectors
- new idea: apply TRAAM, a distributed representation for ITGs
  - we began developing for statistical MT (Addaiki & Wu, 2014)
  - bilingual recursive neural network model
  - uses both input & output language contexts

modeling recursive structures

- TRAAM goes beyond neural network approaches that model monolingual recursive structures
  - neural language models and SRNs (Bengio et al., 2003)
    - contextual history modeled by a RNN
  - convolutional networks (Collobert & Weston, 2008)
    - learn vector representations of words
      - used in NLP tasks such as POS tagging, chunking and SRL
  - RAAs and recursive autoencoders (RAEs) (Palakor, 1999, Socher et al., 2011)
    - can be more flexible than convolutional networks
    - RAAs have been successfully applied to sentiment prediction

bilingual vector space models

- recurrent INLMs model with SMT word alignments
- only learns non-compositional features
**TRAAM model training**
- TRAAM network contains a compressor and a reconstructor network.
  - generalizes RAAM to represent **bilingual** sequences.
  - for **bracketing** ITGs, reordering can be represented by a single bit (straight vs. inverted).
- biparses from a BITG are used to learn the network weights.
  - compressor network generates feature vectors recursively.
  - objective – to ensure the context of children is captured efficiently.
  - reconstructor network provides the loss function.
- backpropagation with structure is used to compute gradients.
  - L-BFGS can be used to optimize network weights (Goller & Kutcher 1996).

**TRAAM forward propagation**
- TRAAM = Compressor
- Computation order: straight
- Compute error δ

**TRAAM backpropagation**
- TRAAM = Reconstructor

**challenges in flamenco**
- no clear boundary between music and dance.
- “constrained improvisation”
- regular and irregular hypermetrical structures.
- rapid switching between 3/4 and 6/8 meters.
- heavy syncopation.
- sudden, misleading off-beat accents and patterns.
- frequent eliding of downbeat accents (which humans and automatic meter-finding algorithms typically rely on).

**learning flamenco hypermetrical structure**
- simultaneous learning of.
  - metrical structure.
  - hypermetrical structure.
  - multipart structural relations.
- learn the relationship between parallel frames of reference.

---

(Wu, ISMIR 2013, SMPC 2015)
learning flamenco hypermetrical structure

initial transduction grammar is just the parallel training corpus

1. $S \rightarrow A$
2. $A \rightarrow A_A$
3. $A \rightarrow D_E$
4. $C \rightarrow A_D$
5. $S \rightarrow A_A$
6. $A \rightarrow A_A$
7. $A \rightarrow B_C$
8. $A \rightarrow D_E$
9. $C \rightarrow A_D$
10. $S \rightarrow A_A$

learning flamenco hypermetrical structure

initial transduction grammar is just the parallel training corpus

1. $S \rightarrow A$
2. $A \rightarrow A_A$
3. $A \rightarrow D_E$
4. $C \rightarrow A_D$
5. $S \rightarrow A_A$
6. $A \rightarrow A_A$
7. $A \rightarrow B_C$
8. $A \rightarrow D_E$
9. $C \rightarrow A_D$
10. $S \rightarrow A_A$

learning flamenco hypermetrical structure

initial transduction grammar is just the parallel training corpus

1. $S \rightarrow A$
2. $A \rightarrow A_A$
3. $A \rightarrow D_E$
4. $C \rightarrow A_D$
5. $S \rightarrow A_A$
6. $A \rightarrow A_A$
7. $A \rightarrow B_C$
8. $A \rightarrow D_E$
9. $C \rightarrow A_D$
10. $S \rightarrow A_A$
learning flamenco hypermetrical structure
The same exact core capability of transduction grammar induction and application appears effective not only for conventional AI tasks, but also for all sorts of creative tasks.