Unsupervised Discovery of Rhyme Schemes

Sravana Reddy
The University of Chicago

Kevin Knight
USC/ISI

June 21, 2011
Motivation

All swol’n with chafing, down Adonis sits,
Banning his boisterous and unruly beast:
And now the happy season once more fits,
That love-sick Love by pleading may be blest;
For lovers say, the heart hath treble wrong
When it is barr’d the aidance of the tongue.

- Shakespeare, 1593

Pronunciations change over time
Stiff, strange, and quaintly coloured
As the broidery of Bayeux
The England of that dawn remains,
And this of Alfred and the Danes
Seems like the tales a whole tribe feigns
Too English to be true.

- Chesteron, 1911
Motivation

Pronunciations may be unknown and not derivable from spelling

- Wang Mian, c. 1300

層樓危構出層霄，把酒登臨客恨饒。
草色不羨吳地短，雁聲空落楚天遙。
江山如畫知豪傑，風月無私慰寂寥。
六代繁華在何處？敗紅殘緑野蕭蕭。
Motivation

Therefore,

we want a language-independent method of finding rhymes that does not need pronunciation information

But

why do we care about finding rhymes anyway?
Motivation

Rhyme scheme annotations are useful –

- Machine Translation of Poetry (Genzel et al., 2010)
  - Rather than dictionary pronunciations (unreliable), train on annotated data

- Digital Humanities (Google Books N-Grams, Perseus Library)
  - Track frequencies, usage trends of rhymes in a large corpus
  - Analyze rhyming word choices of a given poet, etc.

- Historical Linguistics
  - Reconstruct pronunciations from rhymes
    - blest rhymes with beast → cue to how Shakespeare spoke!
Main Cue: Repetition of Rhyming Pairs

sits  tongue  tongue
beast commander so
fits wrong wrong
blest slander show
wrong yet owe
wrong wit surmise eyes
tongue

me  she  me
mine collatine is
infamy me shine
pine mine is mine
Model of Stanza Generation

- Pick a rhyme scheme \( r_1 r_2 \ldots r_n \)
- For \( i \) from 1 to \( n \):
  - If \( r_i = r_j = r_k = \ldots \) for \( j, k, \ldots < i \):
    - Generate word \( w_i \) with probability \( P(w_i | w_j) P(w_i | w_k) \ldots \)
  - Else:
    - Generate word \( w_i \) with prob. \( P(w_i) \)

\[
P(ababbcc) \* P(tongue) \* P(so) \* P(wrong|tongue) \* P(show|so) \* P(owe|so)P(owe|show) \* P(surmise) \* P(eyes|surmise) = P(rhyme scheme) \* (P(stanza|rhyme scheme))
\]
Learning Algorithm

- Find **maximum likelihood rhyme scheme** $r$ for stanza $x$

- Unknown parameters:

  $\theta_{a,b} = \text{strength of ‘rhymingness’ between word } a \text{ and } b$

  $\rho_r = \text{prior probability of rhyme scheme } r$

- Probability of rhyming $a$ with $b$

  $= P(a \mid b) = \frac{\theta_{a,b}}{\sum_c \theta_{c,b}}$

- Let search space for $r = \text{all rhyme schemes in the corpus}$
Expectation Maximization

- **Initialize**: \( \theta_{x, y} \) and \( \rho_r \)

- **E**: posterior probability of rhyme scheme for each stanza.

  \[
P(\text{rhyme scheme } r \mid \text{stanza } x) \text{ under } \theta \text{ and } \rho
  \]

- **M**: Soft counts of rhymingness and prior probabilities

  \[
  \theta_{a, b} = \sum_{x, r: a \text{ rhymes with } b} P(r \mid x) \\
  \rho_r = \sum_x P(r \mid x) / \sum_{x, q} P(q \mid x)
  \]
Orthographic Cues

Initialization of $\theta$

1. Uniform

2. Orthographic Similarity:

$$\theta_{a,b} = \frac{\text{# letters in } a \text{ and } b}{\min (\text{length of } a, \text{length of } b)}$$
Data

- Corpus of manually annotated rhyming poetry
  
  **English:**
  - Time period: 1450-1950
  - 11613 stanzas, 93030 lines
  
  **French:**
  - Time period: 1450-1650
  - 2814 stanzas, 26543 lines

  *From Sonderegger (2011), expanded and edited by us*

  *Collected for this project*
Data

- # of rhyme schemes per stanza length (search space)
Evaluation

- Rhyme Scheme **Accuracy**
- Average **F-Score**

For each word token, look at set of words that rhyme according to gold standard and inferred rhyme scheme.

Compute precision and recall; average F-Score over all tokens.
Results

Rhyme Scheme Accuracy (%)

- English, 1750-1850
- English, 1650-1750
- English, 1550-1650
- Shakespeare
- All English
- All French

Not enough repetition
Results

- **English, 1750-1850**
- **English, 1650-1750**
- **English, 1550-1650**
- **Shakespeare**
- **All English**
- **All French**

- **Naïve Baseline**
- **EM, ortho sim init**

Average F-Score
## Results

- **Comparison with using rhyming definition + CELEX**

| Time Period | Rhymes found by Model | Rhymes found by definition |
|-------------|-----------------------|----------------------------|
| 1450-1550   | left/craft, shone/done| edify/lie, adieu/hue       |
| 1550-1650   | speak/break, doe/two  | obtain/vain, breed/heed    |
| 1650-1750   | most/cost, presage/rage| see/family, blade/shade    |
| 1750-1850   | it/basket, o’er/shore | ice/vice, head/tread       |
| 1850-1950   | of/love, again/rain   | old/enfold, within/win     |
Stanza Dependencies

- This model generates each stanza independently

- But there are connections across stanzas

My mother's maids, when they did sew and spin,
They sang sometime a song of the field mouse
That, for because her livelihood was but thin,
Would needs go seek her townish sister's house.
She thought herself endured too much pain;
The stormy blasts her cave so sore did souse

Wyatt, c. 1500
Stanza Dependencies

- **Solution:** Assume Markov dependencies (each stanza is only related to previous)

- Generative model of stanzas \( x^1 \ x^2 \ x^3 \ldots \ x^m \)

- Generate scheme \( r^1 \)
  and stanza \( x^1 \) as before

For \( i = 2 \) to \( m \)

- Pick rhyme scheme \( r^i \) with prob. \( P(r^i | r^{i-1}) \)
- Generate stanza \( x^i \) with prob. \( P(x^i | r^i, x^{i-1}) \)
Stanza Dependencies

Rhyme Schemes

\[ r^1 \] \[ r^2 \] \[ r^3 \]

Stanzas

\[ x^1 \] \[ x^2 \] \[ x^3 \]

**Autoregressive HMM**

E-Step: compute posteriors with forward-backward algorithm

M-Step: update \( \theta, \rho \)
Results

Rhyme Scheme Accuracy (%)

- Naïve Baseline
- EM, ortho sim init
- EM, stanza dependencies

- English, 1750-1850
- English, 1650-1750
- English, 1550-1650
- All English
- All French
Future Work

- Make use of rhyme transitivity
- Use orthographic similarity and/or rhyming definitions to regularize $\theta$
- Text normalization – infer that
  
  speake/weake = speak/weak
  speaking/weaking = speak/weak
- Incorporate partial supervision when available
- Test on other languages = collect and annotate more data!
Conclusion

- Introduced the problem of unsupervised rhyme scheme annotation

- Solutions using generative models of stanza and rhyme scheme creation

- Outperforms baseline, marked improvements over using pronunciation information for pre-1800 text

- Annotated data and rhyme scheme discovery code in Python available on the ACL Anthology/ACL 2011 proceedings
Thanks for your grace in this chase.