**LI: Supervised Approaches**

**Input**
- A document $d$
- A fixed set of classes $C = \{c_1, c_2, \ldots, c_n\}$
- A training set of $m$ hand-labeled documents $(d_1, c_1), \ldots, (d_m, c_m)$
LI: Supervised Approaches

Input

- A document $d$
- A fixed set of classes $C = \{c_1, c_2, \ldots, c_n\}$
- A training set of $m$ hand-labeled documents $(d_1, c_1), \ldots, (d_m, c_m)$

Output

A learned classifier $\gamma: d \rightarrow c$
Bayes’ rule for documents and classes

For a document $d$ and a class $c$

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$
Bayes’ rule for documents and classes

For a document $d$ and a class $c$

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

Naïve Bayes Classifier

$$c_{MAP} = \arg\max_{c \in C} P(c|d)$$

$$= \arg\max_{c \in C} P(d|c)P(c)$$

$$= \arg\max_{c \in C} P(x_1, x_2, \ldots, x_n|c)P(c)$$
Naïve Bayes classification assumptions

\[ P(x_1, x_2, \ldots, x_n | c) \]
Naïve Bayes classification assumptions

\[ P(x_1, x_2, \ldots, x_n | c) \]

**Bag of words assumption**

Assume that the position of a word in the document doesn’t matter.
Naïve Bayes classification assumptions

\[ P(x_1, x_2, \ldots, x_n | c) \]

**Bag of words assumption**
Assume that the position of a word in the document doesn’t matter

**Conditional Independence**
Assume the feature probabilities \( P(x_i | c_j) \) are independent given the class \( c_j \).

\[ P(x_1, x_2, \ldots, x_n | c) = P(x_1 | c) \cdot P(x_2 | c) \cdots P(x_n | c) \]
Naïve Bayes classification assumptions

\[ P(x_1, x_2, \ldots, x_n | c) \]

**Bag of words assumption**
Assume that the position of a word in the document doesn’t matter

**Conditional Independence**
Assume the feature probabilities \( P(x_i | c_j) \) are independent given the class \( c_j \).

\[ P(x_1, x_2, \ldots, x_n | c) = P(x_1 | c) \cdot P(x_2 | c) \ldots P(x_n | c) \]

\[ c_{NB} = \arg \max_{c \in C} P(c) \prod_{x \in X} P(x | c) \]
Learning the model parameters

Maximum Likelihood Estimate

\[
\hat{P}(c_j) = \frac{doc - count(C = c_j)}{N_{doc}} \\
\hat{P}(w_i|c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}
\]
Learning the model parameters

**Maximum Likelihood Estimate**

\[
\hat{P}(c_j) = \frac{doc - count(C = c_j)}{N_{doc}}
\]

\[
\hat{P}(w_i|c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}
\]

**Problem with MLE**

Suppose in the training data, we haven’t seen one of the words (say *pure*) in a given language.

\[
\hat{P}(pure|Hindi) = 0
\]
Learning the model parameters

**Maximum Likelihood Estimate**

\[
\hat{P}(c_j) = \frac{doc - \text{count}(C = c_j)}{N_{doc}}
\]

\[
\hat{P}(w_i|c_j) = \frac{\text{count}(w_i, c_j)}{\sum_{w \in V} \text{count}(w, c_j)}
\]

**Problem with MLE**

Suppose in the training data, we haven’t seen one of the words (say pure) in a given language.

\[
\hat{P}(\text{pure}|\text{Hindi}) = 0
\]

\[
c_{NB} = \arg \max_c \hat{P}(c) \prod_{x \in X} \hat{P}(x_i|c)
\]
Laplace (add-1) smoothing

\[
\hat{P}(w_i|c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} (count(w, c) + 1)}
\]

\[
= \frac{count(w_i, c) + 1}{(\sum_{w \in V} count(w, c)) + |V|}
\]
### A worked out example

<table>
<thead>
<tr>
<th>Doc</th>
<th>Words</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training 1</td>
<td>English Wikipedia editor</td>
<td>en</td>
</tr>
<tr>
<td>2</td>
<td>free English Wikipedia</td>
<td>en</td>
</tr>
<tr>
<td>3</td>
<td>Wikipedia editor</td>
<td>en</td>
</tr>
<tr>
<td>4</td>
<td>español de Wikipedia</td>
<td>es</td>
</tr>
<tr>
<td>Test 5</td>
<td>Wikipedia español el</td>
<td>?</td>
</tr>
</tbody>
</table>
A worked out example: No smoothing

<table>
<thead>
<tr>
<th>Doc</th>
<th>Words</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>1                English Wikipedia editor</td>
<td>en</td>
</tr>
<tr>
<td></td>
<td>2                free English Wikipedia</td>
<td>en</td>
</tr>
<tr>
<td></td>
<td>3                Wikipedia editor</td>
<td>en</td>
</tr>
<tr>
<td></td>
<td>4                español de Wikipedia</td>
<td>es</td>
</tr>
<tr>
<td>Test</td>
<td>5                Wikipedia español el</td>
<td>?</td>
</tr>
</tbody>
</table>

\[
\hat{P}(c) = \frac{\text{count}(c)}{\sum_{c_j \in C} \text{count}(c_j)}
\]

\[
\hat{P}(t \mid c) = \frac{\text{count}(t,c)}{\sum_{t_i \in V} \text{count}(t_i,c)}
\]

\[P(\text{en}) = \frac{3}{4} \quad P(\text{sp}) = \frac{1}{4} \]

\[P(\text{“Wikipedia”} \mid \text{en}) = \frac{3}{8} \quad P(\text{“Wikipedia”} \mid \text{es}) = \frac{1}{3} \]

\[P(\text{“español”} \mid \text{en}) = \frac{0}{8} \quad P(\text{“español”} \mid \text{es}) = \frac{1}{3} \]

\[P(\text{“el”} \mid \text{en}) = \frac{0}{8} \quad P(\text{“el”} \mid \text{es}) = \frac{0}{3} \]

\[P(\text{en}\mid \text{doc5}) = \frac{3}{4} \times \frac{3}{8} \times \frac{0}{8} = 0 \]

\[P(\text{es}\mid \text{doc5}) = \frac{1}{4} \times \frac{2}{9} \times \frac{1}{3} \times \frac{0}{3} = 0 \]
A worked out example: Smoothing

<table>
<thead>
<tr>
<th>Doc</th>
<th>Words</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>English Wikipedia editor</td>
<td>en</td>
</tr>
<tr>
<td>2</td>
<td>free English Wikipedia</td>
<td>en</td>
</tr>
<tr>
<td>3</td>
<td>Wikipedia editor</td>
<td>en</td>
</tr>
<tr>
<td>4</td>
<td>español de Wikipedia</td>
<td>sp</td>
</tr>
<tr>
<td>5</td>
<td>Wikipedia español el</td>
<td>?</td>
</tr>
</tbody>
</table>

\[
\hat{P}(c) = \frac{\text{count}(c)}{\sum_{c_j \in C} \text{count}(c_j)}
\]

\[
\hat{P}(t | c) = \frac{\text{count}(t, c)}{\sum_{t_i \in V} \text{count}(t_i, c)}
\]

\[
P(\text{en}) = \frac{3}{4} \quad P(\text{sp}) = \frac{1}{4}
\]

\[
P(\text{"Wikipedia" | en}) = 3 + 1/8 + 6 \quad P(\text{"Wikipedia" | sp}) = 1 + 1/3 + 6
\]

\[
P(\text{"español" | en}) = 0 + 1/8 + 6 \quad P(\text{"español" | sp}) = 1 + 1/3 + 6
\]

\[
P(\text{"el" | en}) = 0 + 1/8 + 6 \quad P(\text{"el" | sp}) = 0 + 1/3 + 6
\]

\[
P(\text{en | doc5}) = \frac{3}{4} \times \frac{4}{14} \times \frac{1}{14} \times \frac{1}{14} = 0.00109
\]

\[
P(\text{sp | doc5}) = \frac{1}{4} \times \frac{2}{9} \times \frac{2}{9} \times \frac{1}{9} = 0.00137
\]
Character n-gram based Approach

**Input:** A word \( w \) (e.g., khiprata)
Character n-gram based Approach

**Input:** A word \( w \) (e.g., *kiprata*)

**Features:** character n-grams (n=2 to 5)
Character n-gram based Approach

**Input:** A word $w$ (e.g., khiprata)

**Features:** character n-grams (n=2 to 5)

$khiprata \rightarrow \$kshiprata\$

2: $k, ks, sh, hi, ip, pr, ra, at, ta, a$
3: $ks, ksh, shi, hip, ipr, \ldots\ ta$
4: $ksh, kshi, ship, \ldots, ata$
5: $kshi, kship, shipr, \ldots, rata$
Character n-gram based Approach

**Input:** A word $w$ (e.g., *khiprata*)

**Features:** character n-grams (n=2 to 5)

**Classifier:** Naïve Bayes, Max-Ent, SVMs

$khiprata \rightarrow \$kshiprata\$

2: $k, ks, sh, hi, ip, pr, ra, at, ta, a$
3: $ks, ksh, shi, hip, ipr, ... ta$
4: $ksh, kshi, ship, ..., ata$
5: $kshi, kship, shipr, ..., rata$
Character n-gram based Approach

Input: A word $w$ (e.g., khiprata)

Features: character n-grams (n=2 to 5)
Classifier: Naïve Bayes, Max-Ent, SVMs
Prob (kshiprata is Sanskrit) $\Rightarrow$ Prob (kshiprata is English)
<table>
<thead>
<tr>
<th>Tool</th>
<th>Author</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>langid.py</td>
<td>Lui and Baldwin</td>
<td>[2012]</td>
</tr>
<tr>
<td>ChromeCLD</td>
<td>McCandless</td>
<td>[2010]</td>
</tr>
<tr>
<td>LangDetect</td>
<td>Nakatani</td>
<td>[2010]</td>
</tr>
<tr>
<td>LDIG</td>
<td>Nakatani</td>
<td>[2012]</td>
</tr>
<tr>
<td>whatlang</td>
<td>Brown</td>
<td>[2013]</td>
</tr>
<tr>
<td>YALI</td>
<td>Majliš</td>
<td>[2012]</td>
</tr>
<tr>
<td>TextCat</td>
<td>Scheelen</td>
<td>[2003]</td>
</tr>
<tr>
<td>MSR-LID</td>
<td>Goldszmidt et al.</td>
<td>[2013]</td>
</tr>
</tbody>
</table>
Using langid.py

https://github.com/saffsd/langid.py
Supports 97 languages
### Word-level Language Labeling

Given: \( X = \text{Modi}, \text{ke}, \text{speech}, \text{se}, \text{India}, \text{inspired}, \text{ho}, \text{gaya}, \text{#namo} \)

<table>
<thead>
<tr>
<th>NE</th>
<th>Hn</th>
<th>En</th>
<th>Hn</th>
<th>NE</th>
<th>En</th>
<th>Hn</th>
<th>Hn</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>के</td>
<td>से</td>
<td>हो</td>
<td>गया</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Pawan Goyal (IIT Kharagpur)  NLP for Social Media: Language Identification II  August 3-4, 2016  14 / 48
### Word-level Language Labeling

#### Modeling as a Sequence Prediction Problem

Given \( \mathbf{X} : X_1 = Modi, X_2 = ke, \ldots \)

<table>
<thead>
<tr>
<th>Modi</th>
<th>ke</th>
<th>speech</th>
<th>se</th>
<th>India</th>
<th>inspired</th>
<th>ho</th>
<th>gaya</th>
<th>#namo</th>
</tr>
</thead>
<tbody>
<tr>
<td>NE</td>
<td>Hn</td>
<td>En</td>
<td>Hn</td>
<td>NE</td>
<td>En</td>
<td>Hn</td>
<td>Hn</td>
<td>Other</td>
</tr>
<tr>
<td>के</td>
<td>से</td>
<td>हो</td>
<td>गया</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Pawan Goyal (IIT Kharagpur)  NLP for Social Media: Language Identification II  August 3-4, 2016  14 / 48
Word-level Language Labeling

Modeling as a Sequence Prediction Problem

Given \( X: X_1 = Modi, X_2 = ke, \ldots \)
Output: \( Y = Y_1 \) (label for \( X_1 \)), \( Y_2 \) (label for \( X_2 \)), \ldots
Word-level Language Labeling

Modeling as a Sequence Prediction Problem

Given \( X: X_1 = Modi, X_2 = ke, \ldots \)
Output: \( Y = Y_1 \) (label for \( X_1 \)), \( Y_2 \) (label for \( X_2 \)),\ldots
Such that \( p(Y|X) \) is maximized
Conditional Random Fields: Modelling the Conditional Distribution

Model the Conditional Distribution:

\[ P(y \mid x) \]

To predict a sequence compute:

\[ y^* = \arg \max_y P(y \mid x) \]

Must be able to compute it efficiently.
Conditional Random Fields: Feature Functions

$x_1 \quad x_2 \quad x_3 \quad x_{n-1} \quad x_n$

$y_1 \quad y_2 \quad y_3 \quad y_{n-1} \quad y_n$

Feature Functions:

$f_j(y_{i-1}, y_i, x, i)$
Express some characteristic of the empirical distribution that we wish to hold in the model distribution

\[ f_j(y_{i-1}, y_i, x, i) \]

1. if \( y_{i-1} = \text{IN} \) and \( y_i = \text{NNP} \) and \( x_i = \text{September} \)

0. otherwise
Conditional Random Fields: Distribution

Label sequence modelled as a normalized product of feature functions:

\[
P(y \mid x, \lambda) = \frac{1}{Z(x)} \exp \sum_{i=1}^{n} \sum_{j} \lambda_j f_j (y_{i-1}, y_i, x, i)
\]

\[
Z(x) = \sum_{y \in Y} \sum_{i=1}^{n} \sum_{j} \lambda_j f_j (y_{i-1}, y_i, x, i)
\]
Features for word level Language Identification

Token-based features
- Capitalization
- Script
- Special Characters
- Character n-gram based classifiers
- Word length

Lexical Features
- Regular lexicon
- Unigram Frequency
- Entity Lexicon
- Acronym/slang lexicon

Context Features
- Next 3 tokens
- Last 3 tokens
- Current token
- Previous label (Bigram or B)
Characteristics of Text in Social Media

Social media text contains varying levels of “noise” (lexical, syntactic and otherwise), e.g.
Lexical Normalization

Characteristics of Text in Social Media

Social media text contains varying levels of “noise” (lexical, syntactic and otherwise), e.g.

- Tell ppl u luv them cuz 2morrow is truly not promised.
- SUPER BOWL SUNDAY!!! Enjoy yourselves!!! Sunday morning GOODIES R sent out! C U 2Nyt!
- Follow @OFA today for more coverage of the gun violence petition delivery to Congress. #NotBackingDown #EarlyFF
Eisenstein [2013] identified the following possible contributing factors to “badness” in social media text:

- Lack of literacy?
Eisenstein [2013] identified the following possible contributing factors to “badness” in social media text:

- Lack of literacy? *no*
Eisenstein [2013] identified the following possible contributing factors to “badness” in social media text:

- Lack of literacy? *no*
- Length restrictions?
Eisenstein [2013] identified the following possible contributing factors to “badness” in social media text:

- Lack of literacy? *no*
- Length restrictions? *not primarily*
Why is Social Media Text “Bad”?

Eisenstein [2013] identified the following possible contributing factors to “badness” in social media text:

- Lack of literacy? *no*
- Length restrictions? *not primarily*
- Text input method-driven?
Eisenstein [2013] identified the following possible contributing factors to “badness” in social media text:

- Lack of literacy? *no*
- Length restrictions? *not primarily*
- Text input method-driven? *to some degree, yes*
Eisenstein [2013] identified the following possible contributing factors to “badness” in social media text:

- Lack of literacy? *no*
- Length restrictions? *not primarily*
- Text input method-driven? *to some degree, yes*
- Pragmatics (mimicking prosodic effects etc. in speech)?
Eisenstein [2013] identified the following possible contributing factors to “badness” in social media text:

- Lack of literacy? *no*
- Length restrictions? *not primarily*
- Text input method-driven? *to some degree, yes*
- Pragmatics (mimicking prosodics etc. in speech)? *yeeees*

What can be done about it?

**Lexical normalization**

Translate expressions into their canonical form
Lexical normalization

Translate expressions into their canonical form

Issues

- What are the candidate tokens for normalization?
What can be done about it?

**Lexical normalization**
Translate expressions into their canonical form

**Issues**
- What are the candidate tokens for normalization?
- To what degree do we allow normalization?
What can be done about it?

**Lexical normalization**
Translate expressions into their canonical form

**Issues**
- What are the candidate tokens for normalization?
- To what degree do we allow normalization?
- What is the canonical form of a given expression? (e.g., *aint*)
What can be done about it?

Lexical normalization
Translate expressions into their canonical form

Issues
- What are the candidate tokens for normalization?
- To what degree do we allow normalization?
- What is the canonical form of a given expression? (e.g., aint)
- Is lexical normalization always appropriate? (e.g., bro)
**Task Definition**

**One standard definition**

- relative to some standard tokenization

Assumptions/corrolaries of the task definition:
- not possible to normalize in-vocabulary tokens, e.g., their
- not possible to normalize the multiword tokens, e.g., ttyl
- ignore Twitter-specific entities, e.g., obama, #mandela, bit.ly/1iRqm
- assume a unique correct "norm" for each token
**Task Definition**

**One standard definition**

- relative to some standard tokenization
- consider only OOV tokens as candidates for normalization
**Task Definition**

**One standard definition**
- relative to some standard tokenization
- consider only OOV tokens as candidates for normalization
- allow only one-to-one word substitutions

```
i left ACL  cus  im  sickk !  Yuu  better be their  tmrw
↓  ↓  ↓  ↓
i left ACL because I'm sick ! You better be their tomorrow
```
Task Definition

**One standard definition**

- relative to some standard tokenization
- consider only OOV tokens as candidates for normalization
- allow only one-to-one word substitutions

```
i left ACL cus im sickk ! Yuu better be their tmrw
```

```
i left ACL because I'm sick ! You better be their tomorrow
```

**Assumptions/corrolaries of the task definition:**

- not possible to normalize in-vocabulary tokens, e.g. *their*
Task Definition

One standard definition

- relative to some standard tokenization
- consider only OOV tokens as candidates for normalization
- allow only one-to-one word substitutions

Assumptions/corrolaries of the task definition:

- not possible to normalize in-vocabulary tokens, e.g. *their*
- not possible to normalise the multiword tokens, e.g., *ttyl*
Task Definition

One standard definition

- relative to some standard tokenization
- consider only OOV tokens as candidates for normalization
- allow only one-to-one word substitutions

Assumptions/corrolaries of the task definition:

- not possible to normalize in-vocabulary tokens, e.g. *their*
- not possible to normalise the multiword tokens, e.g., *ttyl*
- ignore Twitter-specific entities, e.g., *obama, #mandela, bit.ly/1iRqm*
Task Definition

One standard definition

- relative to some standard tokenization
- consider only OOV tokens as candidates for normalization
- allow only one-to-one word substitutions

Assumptions/corrolaries of the task definition:

- not possible to normalize in-vocabulary tokens, e.g. their
- not possible to normalise the multiword tokens, e.g., ttyl
- ignore Twitter-specific entities, e.g., obama, #mandela, bit.ly/1iRqm
- assume a unique correct “norm” for each token
Spelling Errors

TOMORROW
- Tomorow
- Tommorow
- Tommorrow

Phonetic/Cognitive Errors

Unintentional Errors

Typos or “slip of finger” errors

- Tpmorrow
- Tomrorrow
- Tmorrow
- Tomnorrow
Understanding unintentional spelling errors

**TOMORROW**
- Tomorrow
- Tomorow
- Tommorrow
- Tpmorrow
- Tomrorow
- Tmorrow
- Tomnorrow

- Double letter omission
- Doubling of wrong letter
- Doubling of letter
- Substitution: o → p
- Metathesis: or → ro
- Deletion: o → ε
- Insertion: ε → n

**Phonetic/Cognitive Errors**

**Typos or “slip of finger” errors**
**Edit Distance**

- **Cost of Edit Operations:**
  - Insertion ($\varepsilon \rightarrow c$): 1
  - Deletion ($c \rightarrow \varepsilon$): 1
  - Substitution: ($c \rightarrow c'$): 1 or 2

  **Metathesis** ($cc' \rightarrow c'c$) is either modeled as a single edit operation (cost = 1) or as a deletion-insertion pair ($cc' \rightarrow \varepsilon c' \rightarrow c'c$), and therefore cost of 2.

- **Edit Distance** between two strings $s: c_1c_2c_3...c_n$ and $s': c'_1c'_2c'_3...c'_n$ is defined as the minimum value of the sum of the cost of a sequence of edit operations required to convert $s$ to $s'$.
  - *engine* & *begin*, *elevator* & *evaluator*, *east* & *csar*

- Dynamic Programming Algorithm
What about spelling errors in Social Media?

The shorter $\rightarrow$ the faster
Constraint: understandability

This is an example for Texting language

Other factors: Coolness, group-membership, accommodating
The case of ‘Tomorrow’

- 2moro (9)
- tomoz (25)
- tomoro (12)
- tomrw (5)
- tom (2)
- tomra (2)
- tomorrow (24)
- tomora (4)

- tomm (1)
- tomo (3)
- tomorow (3)
- 2mro (2)
- morrow (1)
- tomor (2)
- tmorro (1)
- moro (1)

Spell-checkers, such as Aspell, perform very poorly on such data (<22%)

Data from (Choudhury et al., 2007)
**Phonetic substitution (phoneme)**

- Psycho → syco, then → den
- Phonetic substitution (syllable)
  - Today → 2day, see → c
- Deletion of vowels
  - Message → mssg, about → abt
- Deletion of repeated characters
  - Tomorrow → tomorrow
Phonetic substitution (phoneme)

psycho → syco, then → den
**Patterns or Compression Operators**

**Phonetic substitution (phoneme)**

psycho → syco, then → den

**Phonetic substitution (syllable)**

...
### Phonetic substitution (phoneme)

<table>
<thead>
<tr>
<th>Original</th>
<th>Substitution</th>
</tr>
</thead>
<tbody>
<tr>
<td>psycho</td>
<td>syco, then</td>
</tr>
<tr>
<td>then</td>
<td>den</td>
</tr>
</tbody>
</table>

### Phonetic substitution (syllable)

<table>
<thead>
<tr>
<th>Original</th>
<th>Substitution</th>
</tr>
</thead>
<tbody>
<tr>
<td>today</td>
<td>2day, see</td>
</tr>
<tr>
<td>see</td>
<td>c</td>
</tr>
</tbody>
</table>
Patterns or Compression Operators

**Phonetic substitution (phoneme)**

psycho → syco, then → den

**Phonetic substitution (syllable)**

today → 2day, see → c

**Deletion of vowels**
**Phonetic substitution (phoneme)**

psycho → syco, then → den

**Phonetic substitution (syllable)**

today → 2day, see → c

**Deletion of vowels**

message → mssg, about → abt
**Patterns or Compression Operators**

**Phonetic substitution (phoneme)**

psycho → syco, then → den

**Phonetic substitution (syllable)**

today → 2day, see → c

**Deletion of vowels**

message → mssg, about → abt

**Deletion of repeated characters**
Patterns or Compression Operators

**Phonetic substitution (phoneme)**
psycho → syco, then → den

**Phonetic substitution (syllable)**
today → 2day, see → c

**Deletion of vowels**
message → mssg, about → abt

**Deletion of repeated characters**
tomorrow → tomorow
Truncation (deletion of tails)
Truncation (deletion of tails)

introduction $\rightarrow$ intro, evaluation $\rightarrow$ eval
Patterns or Compression Operators

**Truncation (deletion of tails)**

introduction → intro, evaluation → eval

**Common Abbreviations**
Patterns or Compression Operators

**Truncation (deletion of tails)**

introduction → intro, evaluation → eval

**Common Abbreviations**

Kharagpur → kgp, text back → tb
Truncation (deletion of tails)

introduction → intro, evaluation → eval

Common Abbreviations

Kharagpur → kgp, text back → tb

Informal pronunciation
Patterns or Compression Operators

**Truncation (deletion of tails)**

introduction → intro, evaluation → eval

**Common Abbreviations**

Kharagpur → kgp, text back → tb

**Informal pronunciation**

going to → gonna
Patterns or Compression Operators

<table>
<thead>
<tr>
<th>Truncation (deletion of tails)</th>
</tr>
</thead>
<tbody>
<tr>
<td>introduction → intro, evaluation → eval</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Common Abbreviations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kharagpur → kgp, text back → tb</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Informal pronunciation</th>
</tr>
</thead>
<tbody>
<tr>
<td>going to → gonna</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Emphasis by repetition</th>
</tr>
</thead>
</table>
Patterns or Compression Operators

**Truncation (deletion of tails)**
introduction → intro, evaluation → eval

**Common Abbreviations**
Kharagpur → kgp, text back → tb

**Informal pronunciation**
going to → gonna

**Emphasis by repetition**
Funny → fuuuunnnnnyyyyyy
Because → cause (informal usage)
Successive Application of Operators

- Because $\rightarrow$ cause (informal usage)
- cause $\rightarrow$ cauz (phonetic substitution)
Successive Application of Operators

- Because → cause (informal usage)
- cause → cauz (phonetic substitution)
- cauz → cuz (vowel deletion)
Categorisation of non-standard words in English Twitter

<table>
<thead>
<tr>
<th>Category</th>
<th>Ratio</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Letter&amp;Number</td>
<td>2.36%</td>
<td>b4 “before”</td>
</tr>
<tr>
<td>Letter</td>
<td>72.44%</td>
<td>shuld “should”</td>
</tr>
<tr>
<td>Number Substitution</td>
<td>2.76%</td>
<td>4 “for”</td>
</tr>
<tr>
<td>Slang</td>
<td>12.20%</td>
<td>lol “laugh out loud”</td>
</tr>
<tr>
<td>Other</td>
<td>10.24%</td>
<td>sucha “such a”</td>
</tr>
</tbody>
</table>

Table: Types of non-standard words in a 449 message sample of English tweets
Categorisation of non-standard words in English Twitter

Most non-standard words in sampled tweets are morphophonemic variations.

<table>
<thead>
<tr>
<th>Category</th>
<th>Ratio</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Letter &amp; Number</td>
<td>2.36%</td>
<td>b4 “before”</td>
</tr>
<tr>
<td>Letter</td>
<td>72.44%</td>
<td>shuld “should”</td>
</tr>
<tr>
<td>Number Substitution</td>
<td>2.76%</td>
<td>4 “for”</td>
</tr>
<tr>
<td>Slang</td>
<td>12.20%</td>
<td>lol “laugh out loud”</td>
</tr>
<tr>
<td>Other</td>
<td>10.24%</td>
<td>sucha “such a”</td>
</tr>
</tbody>
</table>

Table: Types of non-standard words in a 449 message sample of English tweets

Pawan Goyal (IIT Kharagpur) NLP for Social Media: Language Identification II
Token-based Approach (Han and Baldwin, 2011)
Token-based Approach (Han and Baldwin, 2011)

1. Confusion set generation (i.e., find correction candidates)
2. Non-standard word detection (i.e., is the OOV a non-standard word?)
3. Normalisation of a non-standard word (i.e., select the candidate)

... crush da redberry b4 da water ...

down

b4

Confusion
Generation

before
four
be
bore

...
1. Confusion set generation (i.e., find correction candidates)
2. Non-standard word detection (i.e., is the OOV a non-standard word?)
3. Normalisation of a non-standard word (i.e., select the candidate)

... crush da redberry b4 da water ...

b4 \rightarrow \text{Confusion Generation} \rightarrow \text{before four be bore} ...

\begin{tabular}{c c c c}
\text{before} & ? & \text{da water} \\
\text{crush da redberry} & -3 & -2 & -1 \\
\text{four be bore} & +1 & +2 \\
\end{tabular}

\text{Ill-formed word detector} \rightarrow \text{Yes or No}
Token-based Approach (Han and Baldwin, 2011)

1. Confusion set generation (i.e., find correction candidates)
2. Non-standard word detection (i.e., is the OOV a non-standard word?)
3. Normalisation of a non-standard word (i.e., select the candidate)

... crush da redberry b4 da water ...

b4 → Confusion Generation → before, four, be, bore ...

(crush da redberry ? da water)

four, be, bore ...

ill-formed word detector

Yes or No

candidates and context

before, crush, da ...

four, be, redberry, da ...

before 1.2

four, 0.6

be, 0.1

bore, 0.2
Filter out any Twitter-specific tokens (user-mentions, hashtags, RT, etc.) and URLs
Pre-processing

- Filter out any Twitter-specific tokens (user-mentions, hashtags, RT, etc.) and URLs
- Identify all OOV words relative to a standard spelling dictionary (aspell)
Pre-processing

- Filter out any Twitter-specific tokens (user-mentions, hashtags, RT, etc.) and URLs
- Identify all OOV words relative to a standard spelling dictionary (aspell)
- For OOV words, shorten any repetitions of 3+ letters to 2 letters
Generation via edit distance over letters ($T_c$) and phonemes ($T_p$).
Candidate Generation

- Generation via edit distance over letters ($T_c$) and phonemes ($T_p$).
- This allows to generate “earthquake” for words such as *earthquick*.
Candidate Generation

- Generation via edit distance over letters ($T_c$) and phonemes ($T_p$).
- This allows to generate “earthquake” for words such as earthquick.
- Candidates with $T_c \leq 2 \lor T_p \leq 1$ were taken, further filtered using frequency to take the top 10% of candidates.
Detection of Ill-formed words

Detection based on candidate context fitness

- Correct words should fit better with context than substitution candidates
- Incorrect words should fit worse than substitution candidates
Detection of Ill-formed words

Detection based on candidate context fitness
- Correct words should fit better with context than substitution candidates
- Incorrect words should fit worse than substitution candidates

Basic Idea: Use Dependencies from corpus data
An SVM classifier is trained based on dependencies, to indicate candidate context fitness.

<table>
<thead>
<tr>
<th>Ill-formed word in text snippet</th>
<th>Candidate</th>
<th>Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>but I was thinkin movies.</td>
<td>(thinking, ...)</td>
<td>dobj(thinking, movies)</td>
</tr>
<tr>
<td>article poster by ruderrobb: there was</td>
<td>(rattrap, ...)</td>
<td>–</td>
</tr>
</tbody>
</table>
Feature Representation using Dependencies

- Build a dependency bank from existing corpora
- Represent each dependency tuple as a word pair + positional index
Feature Representation using Dependencies

- Build a dependency bank from existing corpora
- Represent each dependency tuple as a word pair + positional index

Corpus (NYT)
One obvious difference is the way they look, ...

Stanford Parser

```
num(difference-3, One-1)
amod(difference-3, obvious-2)
nsubj(way-6, difference-3)
cop(way-6, is-4)
det(way-6, the-5)
dobj(look-8, way-6)
nsubj(look-8, they-7)
rcmod(way-6, look-8)
```

Dependency bank

```
(way, difference, 3)
(look, way, 2)
```

...
SVM Training Data Generation

- Use dependency bank directly as positive features
Use dependency bank directly as positive features

Automatically generate negative dependency features by replacing the target word with highly-ranked confusion candidates
SVM Training Data Generation

- Use dependency bank directly as positive features
- Automatically generate negative dependency features by replacing the target word with highly-ranked confusion candidates

![Diagram showing dependency bank and positive/negative samples](image)

- **Dependency bank**
  - (look, way, +2)
  - (way, difference, +3)
  - ...

- **Positive samples**
  - (look, way, +2)
  - ...

- **Negative samples**
  - (hook, way, +2)
  - ...

- **Context fitness classifier**

Pawan Goyal (IIT Kharagpur)  
NLP for Social Media: Language Identification II  
August 3-4, 2016  
40 / 48
Detecting ill-formed words

OOV words with candidates fitting the context (i.e., positive classification outputs) are probably ill-formed words.
Detecting ill-formed words

OOV words with candidates fitting the context (i.e., positive classification outputs) are probably ill-formed words

...way yu lookin shuld be a sin..

looking ~ (way, looking, -2)
hooking ~ (way, hooking, -2)

If positive outputs exceed the threshold, feed all candidates for normaliation.

Prediction

<table>
<thead>
<tr>
<th>looking</th>
<th>hooking</th>
</tr>
</thead>
<tbody>
<tr>
<td>~ +1</td>
<td>~ -1</td>
</tr>
</tbody>
</table>

...
Detecting ill-formed words

OOV words with candidates fitting the context (i.e., positive classification outputs) are probably ill-formed words

\[
\text{...way yu } \text{lookin} \text{ shuld be a sin ..}
\]

\[
\text{looking } \sim (\text{way}, \text{looking}, -2)
\]

\[
\text{hooking } \sim (\text{way}, \text{hooking}, -2)
\]

\[
\text{Prediction}
\]

\[
\text{Context fitness classifier}
\]

\[
\text{If positive outputs exceed the threshold, feed all candidates for normalization.}
\]

\[
\text{looking } \sim +1
\]

\[
\text{hooking } \sim -1
\]

\[
\text{...}
\]

Threshold = 1 → lookin is considered to be an ill-formed word
For each ill-formed word and its possible correction candidates, the following features are considered for normalization:

**Word Similarity**
- letter and phoneme edit distance (ED)
- prefix, suffix, and longest common subsequence
Normalization Candidate Selection

For each ill-formed word and its possible correction candidates, the following features are considered for normalization:

**Word Similarity**
- letter and phoneme edit distance (ED)
- prefix, suffix, and longest common subsequence

**Context Support**
- trigram language model score
- dependency score (weighted dependency count, derived from the detection step)
**Type-based approach**

**Observation**

The longer the ill-formed word, the more likely there is a unique normalization candidate.
**Type-based approach**

**Observation**

The longer the ill-formed word, the more likely there is a unique normalization candidate

- \( y \Rightarrow \{ \text{why, you, ...} \} \), \( hw \Rightarrow \{ \text{how, homework, ...} \} \)
- \( 4eva \Rightarrow \{ \text{forever} \} \), \( tlkin \Rightarrow \{ \text{talking} \} \)
Type-based approach

Observation
The longer the ill-formed word, the more likely there is a unique normalization candidate

- \( y \Rightarrow \{\text{why, you, ... }\} \)
- \( \text{hw} \Rightarrow \{\text{how, homework, ... }\} \)
- \( \text{4eva} \Rightarrow \{\text{forever}\} \)
- \( tlkin \Rightarrow \{\text{talking}\} \)

Approach
Construct a dictionary of (lexical variant, standard form) pair for longer word types (character length \( \geq 4 \)) of moderate frequency (\( \geq 16 \))
Construct the dictionary based on distributional similarity + string similarity
Type-based Approach (Han et al. (2012))

Construct the dictionary based on distributional similarity + string similarity

**Input: Tokenised English tweets**
- Extract (OOV, IV) pairs based on distributional similarity
- Re-rank the extracted pairs by string similarity
Type-based Approach (Han et al. (2012))

Construct the dictionary based on distributional similarity + string similarity

**Input: Tokenised English tweets**
- Extract (OOV, IV) pairs based on distributional similarity
- Re-rank the extracted pairs by string similarity

**Output**
A list of (OOV, IV) pairs ordered by string similarity; select the top-$n$ pairs for inclusion in the normalisation lexicon.
An Example

... see you tmrw ...
... tmrw morning ...
... tomorrow morning ...
...

↓ distributional similarity

\{tmrw, 2morow, tomorrow, Monday\}

↓ string similarity

\[tmrw \rightarrow tomorrow\]
Context Modelling

Components/parameters of the method

- context window size: ±1, ±2, ±3
- context word sensitivity: bag-of-words vs. positional indexing
- context word representation: unigram, bigram or trigram
- context word filtering: all tokens vs. only dictionary words
- context similarity: KL divergence, Jensen-Shannon divergence, Cosine similarity, Euclidean distance
Components/parameters of the method

- context window size: ±1, ±2, ±3
- context word sensitivity: bag-of-words vs. positional indexing
- context word representation: unigram, bigram or trigram
- context word filtering: all tokens vs. only dictionary words
- context similarity: KL divergence, Jensen-Shannon divergence, Cosine similarity, Euclidean distance

Tune parameters relative to (OOV, IV) pair development data
Rerank pairs by string similarity

(OOV,IV) pairs derived by distributional similarity:

\( (\text{Obama}, \text{Adam}) \downarrow \)
\( (\text{tmrw}, \text{tomorrow}) \uparrow \)
\( (\text{Youtube}, \text{web}) \downarrow \)
\( (\text{4eva}, \text{forever}) \uparrow \)

\ldots
Rerank pairs by string similarity

(OOV,IV) pairs derived by distributional similarity:

\[
(\text{Obama}, \text{Adam}) \downarrow \\
(tmrw, \text{tomorrow}) \uparrow \\
(\text{Youtube}, \text{web}) \downarrow \\
(4eva, \text{forever}) \uparrow \\
\ldots \\
(tmrw, \text{tomorrow}) \\
(4eva, \text{forever})
\]

Get the top-ranked pairs as lexicon entries:
Main References
