NLP for Social Media: Language Identification II and Text Normalization

Pawan Goyal

CSE, IITKGP

August 3-4, 2016

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

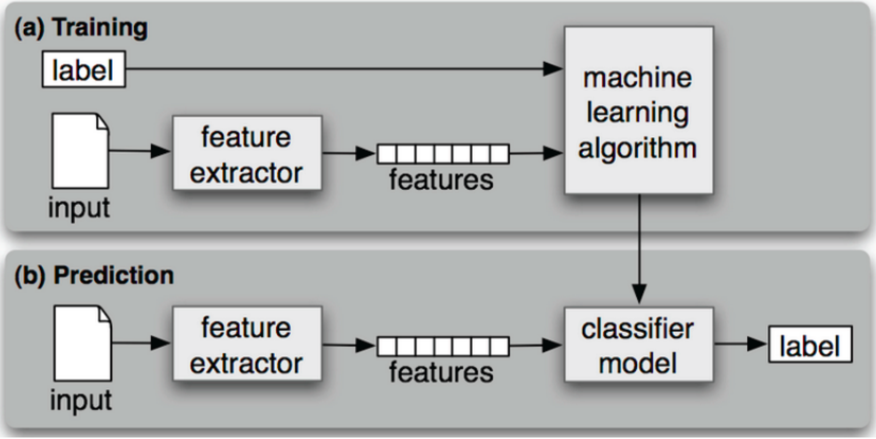
Input

- 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 $\gamma : d \to c$

Supervised Machine Learning



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

$$\begin{aligned} c_{MAP} &= \arg\max_{c \in C} P(c|d) \\ &= \arg\max_{c \in C} P(d|c)P(c) \\ &= \arg\max_{c \in C} P(x_1, x_2, \ldots, x_n|c)P(c) \end{aligned}$$

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.

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

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.

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

$$c_{NB} = \underset{c \in C}{\arg\max}\, P(c) \prod_{x \in X} P(x|c)$$

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

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

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

Problem with MLE

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

$$\hat{P}(pure|Hindi) = 0$$

$$c_{NB} = \arg\max_{c} \hat{P}(c) \prod_{x \in X} \hat{P}(x_i|c)$$

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

	Doc	Words	Class
Training	1	English Wikipedia editor	en
	2	free English Wikipedia	en
	3	Wikipedia editor	en
	4	español de Wikipedia	es
Test	5	Wikipedia español el	?

A worked out example: No smoothing

	Doc	Words	Class
Training	1	English Wikipedia editor	en
	2	free English Wikipedia	en
	3	Wikipedia editor	en
	4	español de Wikipedia	es
Test	5	Wikipedia español el	?

$$\hat{P}(c) = \frac{count(c)}{\sum_{c_j \in C} count(c_j)}$$

P(en)=3/4 **P(sp)=1/4**

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

P("Wikipedia" | en) = 3/8 , P("Wikipedia" | es) = 1/3
P("español" | en) = 0/8 , P("español" | es) = 1/3
P("el" | en) = 0/8 , P("el" | es) = 0/3

P(en | doc5) = 3/4×3/8×0/8×0/8 = 0
P(es | doc5) = 1/4×2/9×1/3×0/3 = 0

A worked out example: Smoothing

	Doc	Words	Class
Training	1	English Wikipedia editor	en
	2	free English Wikipedia	en
	3	Wikipedia editor	en
	4	español de Wikipedia	sp
Test	5	Wikipedia español el	?

$$\hat{P}(c) = \frac{count(c)}{\sum_{c_j \in C} count(c_j)}$$

P(en)=3/4 P(sp)=1/4

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

P("Wikipedia" |en) = 3+1/8+6 , P("Wikipedia" |sp) = 1+1/3+6
P("español" |en) = 0+1/8+6 , P("español" |sp) = 1+1/3+6
P("el" |en) = 0+1/8+6 , P("el" |sp) = 0+1/3+6

P(en|doc5) = 3/4×4/14×1/14×1/14 = 0.00109
P(sp|doc5) = 1/4×2/9×2/9×1/9 = 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., *khiprata*)

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)

kshiprata → $kshiprata$
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*)

kshiprata → $kshiprata$
2: $k, ks, sh, hi, ip, pr, ra, at, ta, a$
3: $ks, ksh, shi, hip, ipr, ... ta$
4: $ksh, kshi, ship, ..., ata$
5: $kshi, kship, shipr, ..., rata$

Features: character n-grams (n=2 to 5)
Classifier: Naïve Bayes, Max-Ent, SVMs

Character n-gram based Approach

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

kshiprata → $kshiprata$

2: $k, ks, sh, hi, ip, pr, ra, at, ta, a$

3: $ks, ksh, shi, hip, ipr, ... ta$

4: $ksh, kshi, ship, ..., ata$

5: $kshi, kship, shipr, ..., rata$

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

Classifier: Naïve Bayes, Max-Ent, SVMs

Prob (kshiprata is Sanskrit) » Prob (kshiprata is English)

LangID Tools

`langid.py`	Lui and Baldwin [2012]
`ChromeCLD`	McCandless [2010]
`LangDetect`	Nakatani [2010]
`LDIG`	Nakatani [2012]
`whatlang`	Brown [2013]
`YALI`	Majliš [2012]
`TextCat`	Scheelen [2003]
`MSR-LID`	Goldszmidt et al. [2013]

Using langid.py

```
python
Python 2.7.2+ (default, Oct  4 2011, 20:06:09)
[GCC 4.6.1] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import langid
>>> langid.classify("I do not speak english")
('en', 0.57133487679900674)
>>> langid.set_languages(['de','fr','it'])
>>> langid.classify("I do not speak english")
('it', 0.99999835791478453)
>>> langid.set_languages(['en','it'])
>>> langid.classify("I do not speak english")
('en', 0.99176190378750373)
```

https://github.com/saffsd/langid.py

Supports 97 languages

Word-level Language Labeling

Modi	ke	speech	se	India	inspired	ho	gaya	#namo
NE	Hn	En	Hn	NE	En	Hn	Hn	Other
	के		से			हो	गया	

Word-level Language Labeling

Modi	ke	speech	se	India	inspired	ho	gaya	#namo
NE	Hn	En	Hn	NE	En	Hn	Hn	Other
	के		से			हो	गया	

Modeling as a Sequence Prediction Problem

Given **X:** $X_1 = Modi, X_2 = ke, \ldots$

Word-level Language Labeling

Modi	ke	speech	se	India	inspired	ho	gaya	#namo
NE	Hn	En	Hn	NE	En	Hn	Hn	Other
	के		से			हो	गया	

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),...

Word-level Language Labeling

Modi	ke	speech	se	India	inspired	ho	gaya	#namo
NE	Hn	En	Hn	NE	En	Hn	Hn	Other
	के		से			हो	गया	

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),...

Such that $p(Y|X)$ is maximized

Conditional Random Fields: Modelling the Conditional Distribution

Conditional Random Fields: Modelling the Conditional Distribution

Model the Conditional Distribution:

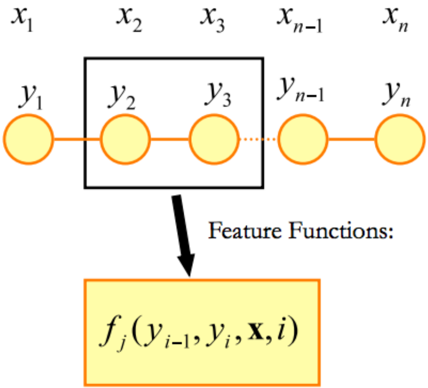
$$P(\mathbf{y} \mid \mathbf{x})$$

To predict a sequence compute:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x})$$

Must be able to compute it efficiently.

Conditional Random Fields: Feature Functions



Feature Functions

Express some characteristic of the empirical distribution that we wish to hold in the model distribution

$$f_j(y_{i-1}, y_i, \mathbf{x}, i) \rightarrow \begin{cases} 1 \quad \text{if } y_{i-1} = IN \text{ and } \\ \qquad y_i = NNP \text{ and } \\ \qquad x_i = September \\ 0 \quad \text{otherwise} \end{cases}$$

Conditional Random Fields: Distribution

Label sequence modelled as a normalized product of feature functions:

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

$$Z(\mathbf{x}) = \sum_{\mathbf{y} \in Y} \sum_{i=1}^{n} \sum_{j} \lambda_j f_j(y_{i-1}, y_i, \mathbf{x}, i)$$

Features for word level Language Identification

Token-based features	Lexical Features	Context Features
• Capitalization • Script • Special Characters • Character n-gram based classifiers • Word length	• Regular lexicon • Unigram Frequency • Entity Lexicon • Acronym/slang lexicon	• Next 3 tokens • Last 3 tokens • Current token • Previous label (Bigram or B)

Lexical Normalization

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

Why is Social Media Text "Bad"?

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

- Lack of literacy?

Why is Social Media Text "Bad"?

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

- Lack of literacy? *no*

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?

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*

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?

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? *to some degree, yes*

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? *to some degree, yes*
- Pragmatics (mimicking prosodic effects etc. in speech)?

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? *to some degree, yes*
- Pragmatics (mimicking prosodic effects etc. in speech)? *yeeees*

Eisenstein, *What to do about bad language on the internet*, NAACL-HLT, 2013

What can be done about it?

Lexical normalization

Translate expressions into their canonical form

What can be done about it?

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

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

$$
\begin{array}{cccccccccccc}
\textit{i left ACL} & \underline{\textit{cus}} & \underline{\textit{im}} & \underline{\textit{sickk}} & ! & \underline{\textit{Yuu}} & \textit{better be their} & \underline{\textit{tmrw}} \\
 & \Downarrow & \Downarrow & \Downarrow & & \Downarrow & & \Downarrow \\
\textit{i left ACL} & \underline{\textit{because}} & \underline{\textit{I'm}} & \underline{\textit{sick}} & ! & \underline{\textit{You}} & \textit{better be their} & \underline{\textit{tomorrow}}
\end{array}
$$

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

$\Downarrow$

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

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

$\Downarrow$

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*
- 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

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*
- 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

$$\begin{array}{cccccccc} \text{i left ACL} & \underline{\text{cus}} & \underline{\text{im}} & \underline{\text{sickk}} & ! & \underline{\text{Yuu}} & \text{better be their} & \underline{\text{tmrw}} \\ & \Downarrow & \Downarrow & \Downarrow & & \Downarrow & & \Downarrow \\ \text{i left ACL} & \underline{\text{because}} & \underline{\text{I'm}} & \underline{\text{sick}} & ! & \underline{\text{You}} & \text{better be their} & \underline{\text{tomorrow}} \end{array}$$

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

- Tpmorrow
- Tomrorow
- Tmorrow
- Tomnorrow

Typos or “slip of finger” errors

Unintentional Errors

Understanding unintentional spelling errors

TOMORROW

- Tomorow — Double letter omission
- Tommorow — Doubling of wrong letter
- Tommorrow — Doubling of letter

Phonetic/Cognitive Errors

- Tpmorrow — Substitution: o→ p
- Tomrorow — Metathesis: or→ ro
- Tmorrow — Deletion: o→ ε
- Tomnorrow — Insertion: ε → n

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**:$c_1 c_2 c_3 \ldots c_n$ and **s'**:$c'_1 c'_2 c'_3 \ldots c'_n$ 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 → the faster
Constraint: understandability

24

dis is n eg 4 txtin lang

39

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)

Patterns or Compression Operators

Phonetic substitution (phoneme)

Patterns or Compression Operators

Phonetic substitution (phoneme)

psycho $\rightarrow$ syco, then $\rightarrow$ den

Patterns or Compression Operators

Phonetic substitution (phoneme)

psycho $\rightarrow$ syco, then $\rightarrow$ den

Phonetic substitution (syllable)

Patterns or Compression Operators

Phonetic substitution (phoneme)

psycho $\rightarrow$ syco, then $\rightarrow$ den

Phonetic substitution (syllable)

today $\rightarrow$ 2day, see $\rightarrow$ c

Patterns or Compression Operators

Phonetic substitution (phoneme)

psycho $\rightarrow$ syco, then $\rightarrow$ den

Phonetic substitution (syllable)

today $\rightarrow$ 2day, see $\rightarrow$ c

Deletion of vowels

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

Patterns or Compression Operators

Phonetic substitution (phoneme)

psycho $\rightarrow$ syco, then $\rightarrow$ den

Phonetic substitution (syllable)

today $\rightarrow$ 2day, see $\rightarrow$ c

Deletion of vowels

message $\rightarrow$ mssg, about $\rightarrow$ abt

Deletion of repeated characters

Patterns or Compression Operators

Phonetic substitution (phoneme)

psycho $\rightarrow$ syco, then $\rightarrow$ den

Phonetic substitution (syllable)

today $\rightarrow$ 2day, see $\rightarrow$ c

Deletion of vowels

message $\rightarrow$ mssg, about $\rightarrow$ abt

Deletion of repeated characters

tomorrow $\rightarrow$ tomorow

Patterns or Compression Operators

Truncation (deletion of tails)

Patterns or Compression Operators

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 $\rightarrow$ intro, evaluation $\rightarrow$ eval

Common Abbreviations

Kharagpur $\rightarrow$ kgp, text back $\rightarrow$ tb

Patterns or Compression Operators

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 $\rightarrow$ intro, evaluation $\rightarrow$ eval

Common Abbreviations

Kharagpur $\rightarrow$ kgp, text back $\rightarrow$ tb

Informal pronunciation

going to $\rightarrow$ gonna

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

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

Successive Application of Operators

- Because $\rightarrow$ cause (informal usage)

Successive Application of Operators

- Because $\rightarrow$ cause (informal usage)
- cause $\rightarrow$ cauz (phonetic substitution)

Successive Application of Operators

- Because $\rightarrow$ cause (informal usage)
- cause $\rightarrow$ cauz (phonetic substitution)
- cauz $\rightarrow$ cuz (vowel deletion)

Categorisation of non-standard words in English Twitter

Category	Ratio	Example
Letter&Number	2.36%	*b4* “before”
Letter	72.44%	*shuld* “should”
Number Substitution	2.76%	*4* “for”
Slang	12.20%	*lol* “laugh out loud”
Other	10.24%	*sucha* “such a”

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

Categorisation of non-standard words in English Twitter

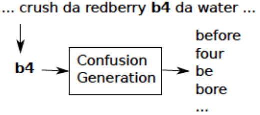
Category	**Ratio**	Example
Letter&Number	2.36%	*b4* "before"
Letter	72.44%	*shuld* "should"
Number Substitution	2.76%	*4* "for"
Slang	12.20%	*lol* "laugh out loud"
Other	10.24%	*sucha* "such a"

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

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

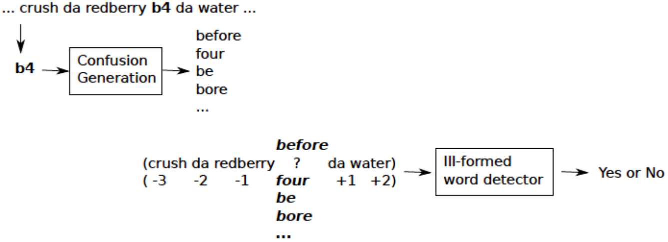
Token-based Approach (Han and Baldwin, 2011)

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)



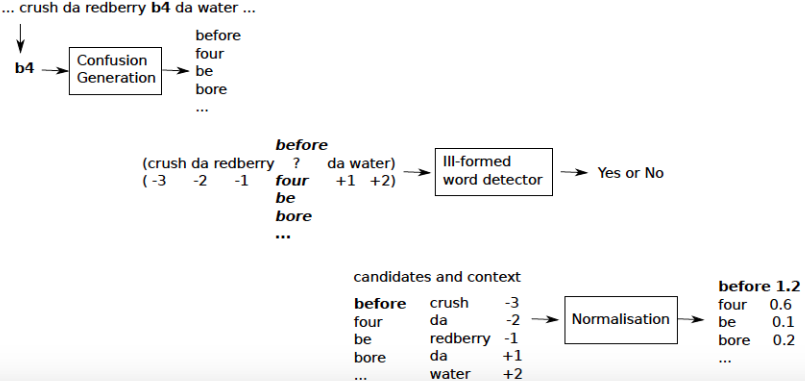
Token-based Approach (Han and Baldwin, 2011)

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)



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)



Pre-processing

- 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

Candidate Generation

- 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 $T_c \leq 2 \vee T_p \leq 1$ 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.

Ill-formed word in text snippet	Candidate	Dependencies
but I was thinkin movies .	(thinking, ...)	dobj(thinking, movies)
article poster by ruderrobb : there was	(rattrap, ...)	–

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

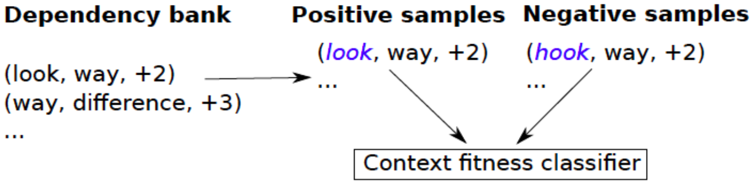
- Use dependency bank directly as positive features

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

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

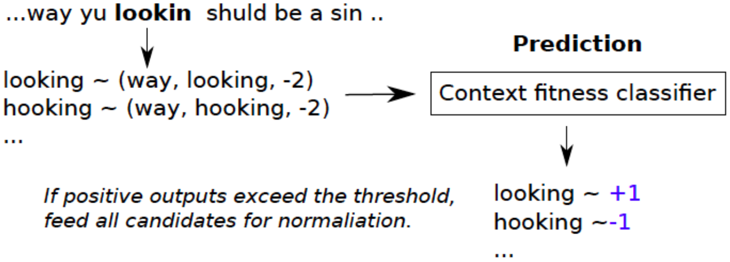


Detecting ill-formed words

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

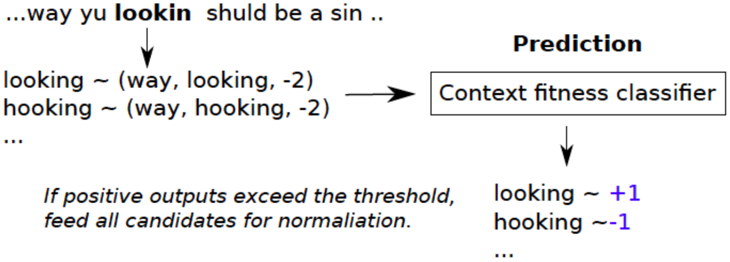
Detecting ill-formed words

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



Detecting ill-formed words

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



Threshold = 1 → *lookin* is considered to be an ill-formed word

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

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

- $\underline{y} \Rightarrow \{\underline{why}, \underline{you}, \ldots\}$, $\underline{hw} \Rightarrow \{\underline{how}, \underline{homework}, \ldots\}$
- $\underline{4eva} \Rightarrow \{\underline{forever}\}$, $\underline{tlkin} \Rightarrow \{\underline{talking}\}$

Type-based approach

Observation

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

- $\underline{y} \Rightarrow \{\underline{why}, \underline{you}, \ldots\}$, $\underline{hw} \Rightarrow \{\underline{how}, \underline{homework}, \ldots\}$
- $\underline{4eva} \Rightarrow \{\underline{forever}\}$, $\underline{tlkin} \Rightarrow \{\underline{talking}\}$

Approach

Construct a dictionary of (lexical variant, standard form) pair for longer word types (character length ≥ 4) of moderate frequency (≥ 16)

Type-based Approach (Han et al. (2012))

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* ...

...

$\Downarrow$ distributional similarity

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

$\Downarrow$ string similarity

tmrw $\rightarrow$ *tomorrow*

Context Modelling

Components/parameters of the method

- context wondow 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

Context Modelling

Components/parameters of the method

- context wondow 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 develepment data

Rerank pairs by string similarity

(OOV,IV) pairs derived by distributional similarity:

$(Obama, Adam)\ \downarrow$
$(tmrw, tomorrow) \uparrow$
$(Youtube, web)\ \downarrow$
$(4eva, forever)\ \uparrow$
$\ldots$

Rerank pairs by string similarity

(OOV,IV) pairs derived by distributional similarity:

$(Obama, Adam) \downarrow$
$(tmrw, tomorrow) \uparrow$
$(Youtube, web) \downarrow$
$(4eva, forever) \uparrow$
$\ldots$

Get the top-ranked pairs as lexicon entries: $(tmrw, tomorrow)$ $(4eva, forever)$

Main References

- Han, Bo, and Timothy Baldwin. "Lexical normalisation of short text messages: Makn sens a# twitter." Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. Association for Computational Linguistics, 2011.
- Han, Bo, Paul Cook, and Timothy Baldwin. "Automatically constructing a normalisation dictionary for microblogs." Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning. Association for Computational Linguistics, 2012.