Text Normalization

Richard Sproat, Steven Bedrick
Text Normalization

• Conversion of text that includes ‘non-standard’ words like numbers, abbreviations, misspellings . . . into normal words.
  – Abbreviation expansion (including novel abbreviations)
  – Expansion of numbers into ‘number names’
  – Correction of misspellings
  – Disambiguation in cases where there is ambiguity
Where is normalization needed?

• Very little in cases like this:

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, ‘and what is the use of a book,’ thought Alice ‘without pictures or conversation?’

So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.
Where is normalization needed?

• A lot in cases like this:

```
CUST RCVD LTTR CNCRNG LOCAL SRVC
VISIT NECESSARY BUT CST STILL HAS PAC BELL SERV ON OLD TN AT RESIDENCE
ORDERD CALLNG CRDS PER CSR RQST
1st att, left mssg for CB from Lynda, will wait for call
50’s Sutton Place Area Convertible 3BR 1400 SF 2BR, 2Bth, L-Shaped LR, S.E.
Open Vus, Gar, Rf Dk, Mid $400K’s Thompson Kane Ina 339-8300
```
Humans are pretty good at this: can you read this?

f u cn rd ths thn u r dng btr thn ny autmtc txt nrmlztn prgrm cn do.
Or this?

Goccdrnia to a hscheearcr at Emabrigdc Yinervtisu, it teosn’d rttaem in tahw rredo the stteerl in a drow are, the ylno tprmoetni gihnt is taht the trisf and tsal rtteel be at the tghir eclap. The tser can be a lotat ssem and you can litls daer it touthisw morbelp. Siht is ecuseab the nuamh dnim seod not daer yrvee rtetel by fstlei, but the drow as a elohw.
Two components of text normalization

• Given a string of characters in a text, what is the (reasonable) set of possible actual words (or word sequences) that might correspond to it.

• Which of those is right for the particular context?
An illustration

He has 123 goats.

I live at King Avenue.
Two components of text normalization

• A component that gives you the set of possibilities:
  – $123 = \text{one hundred (and) twenty three}$
  – $123 = \text{one twenty three}$
  – $123 = \text{one two three}$

• A component that tells you which one(s) are appropriate to a particular context.
A concrete example of finite-state methods in text normalization: digit to number name translation

• Factor digit string:
  – 123 → $1 \cdot 10^2 + 2 \cdot 10^1 + 3$

• Translate factors into number names:
  – $10^2$ → hundred
  – $2 \cdot 10^1$ → twenty
  – $1 \cdot 10^1 + 3$ → thirteen

• Languages vary on how extensive these lexicons are. Some (e.g. Chinese) have very regular (hence very simple) number name systems; others (e.g. Urdu/Hindi) have a large set of number names with a name for almost every number from 1 to 100.

• Each of these steps can be accomplished with FSTs
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>eik</th>
<th>21</th>
<th>ik-kees</th>
<th>41</th>
<th>ikta-lees</th>
<th>61</th>
<th>ik-shat</th>
<th>81</th>
<th>ik-si</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>dau</td>
<td>22</td>
<td>ba-ees</td>
<td>baya-lees</td>
<td>62</td>
<td>ba-shat</td>
<td>82</td>
<td>baya-si</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>teen</td>
<td>23</td>
<td>ta-ees</td>
<td>tainta-lees</td>
<td>63</td>
<td>tere-shat</td>
<td>83</td>
<td>tera-si</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>chaar</td>
<td>24</td>
<td>chau-bees</td>
<td>chawa-lees</td>
<td>64</td>
<td>chaun-shat</td>
<td>84</td>
<td>chaura-si</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>paanch</td>
<td>25</td>
<td>pach-chees</td>
<td>painta-lees</td>
<td>65</td>
<td>paen-shat</td>
<td>85</td>
<td>picha-si</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>chay</td>
<td>26</td>
<td>chab-bees</td>
<td>chaya-lees</td>
<td>66</td>
<td>sar-shat / chay-aa-shat</td>
<td>86</td>
<td>chaya-si</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>saath</td>
<td>27</td>
<td>satta-ees</td>
<td>santa-lees</td>
<td>67</td>
<td>sataath</td>
<td>87</td>
<td>sata-si</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>aath</td>
<td>28</td>
<td>attha-ees</td>
<td>arta-lees</td>
<td>68</td>
<td>athath</td>
<td>88</td>
<td>atha-si</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>nau</td>
<td>29</td>
<td>unat-tees</td>
<td>un-chas</td>
<td>69</td>
<td>unat-tar</td>
<td>89</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>dus</td>
<td>30</td>
<td>tees</td>
<td>pa-chas</td>
<td>70</td>
<td>sat-tar</td>
<td>90</td>
<td>navay</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>gyaa-raan</td>
<td>31</td>
<td>ikat-tees</td>
<td>ika-vun</td>
<td>71</td>
<td>ikat-tar</td>
<td>91</td>
<td>ikan-vay</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>baa-raan</td>
<td>32</td>
<td>bat-tees</td>
<td>ba-vun</td>
<td>72</td>
<td>bahat-tar</td>
<td>92</td>
<td>ban-vay</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>te-raan</td>
<td>33</td>
<td>tain-tees</td>
<td>tera-pun</td>
<td>73</td>
<td>tehat-tar</td>
<td>93</td>
<td>teran-vay</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>chau-daan</td>
<td>34</td>
<td>chaun-tees</td>
<td>chav-van</td>
<td>74</td>
<td>chohat-tar</td>
<td>94</td>
<td>chauran-vay</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>pand-raan</td>
<td>35</td>
<td>pan-tees</td>
<td>pach-pa</td>
<td>75</td>
<td>pagat-tar</td>
<td>95</td>
<td>pichan-vay</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>so-laan</td>
<td>36</td>
<td>chat-tees</td>
<td>chap-pan</td>
<td>76</td>
<td>chayat-tar</td>
<td>96</td>
<td>chiyan-vay</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>sat-raan</td>
<td>37</td>
<td>san-tees</td>
<td>sata-van</td>
<td>77</td>
<td>satat-tar</td>
<td>97</td>
<td>chatan-vay</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>attha-raan</td>
<td>38</td>
<td>ear-tees</td>
<td>atha-van</td>
<td>78</td>
<td>athat-tar</td>
<td>98</td>
<td>athan-vay</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>un-nees</td>
<td>39</td>
<td>unta-tees</td>
<td>un-shat</td>
<td>79</td>
<td>una-si</td>
<td>99</td>
<td>ninan-vay</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>bees</td>
<td>40</td>
<td>cha-tees</td>
<td>shaat</td>
<td>80</td>
<td>assi</td>
<td>100</td>
<td>saw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Digit string factoring transducer (fragment)
Germanic “decade flop”

zwanzig  vier
4  und
70’s
Digit-string to number name translation: German

- **Factor digit string:**
  \[ 123 \rightarrow 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \]

- **Flip decades and units:**
  \[ 2 \cdot 10^1 + 3 \rightarrow 3 + 2 \cdot 10^1 \]

- **Translate factors into number names:**
  \[ 10^2 \rightarrow \text{hundert} \]
  \[ 2 \cdot 10^1 \rightarrow \text{zwanzig} \]
  \[ 1 \cdot 10^1 + 3 \rightarrow \text{dreizehn} \]
German number grammar (fragment)

TEN → 1 \cdot 10^1 \text{ zehn} | \text{TEENW}
TEN → \text{UNITW und TENW}
TEN → \text{UNITW}
TEN → \text{TENW}

TENW → 2 \cdot 10^1 \text{ zwanzig} |
3 \cdot 10^1 \text{ dreißig} |
4 \cdot 10^1 \text{ vierzig} |
5 \cdot 10^1 \text{ fünfzig} . . .

TEENW → 1 \cdot 10^1 + 1 \text{ elf} |
1 \cdot 10^1 + 2 \text{ zwölf} |
1 \cdot 10^1 + 3 \text{ dreizehn} |
1 \cdot 10^1 + 4 \text{ vierzehn} . . .
Concrete example from English

Consider a machine that maps between digit strings and their reading as number names in English.

30,294,005,179,018,903.56 →
thirty quadrillion, two hundred and ninety four trillion, five billion, one hundred seventy nine million, eighteen thousand, nine hundred three, point five six
566 states and 1492 arcs
The Problem: Rampant Abbreviations

- UNE-P RAMP notes:
  CUST RCVD LTTR CNCRNG LOCAL SRVC
  VISIT NECESSARY BUT CST STILL HAS PAC BELL SERV ON OLD TN AT RESIDENCE
  ORDERD CALLNG CRDS PER CSR RQST

- Worldnet notes:
  Cust wanted to know if we currently had 4.95 pp Adv we do not
  cust still has at&t s/w on comp he is going to be moving to PA in a mth and wants to know if
  he can reactivate this acct

- LIFE Remarks:
  1st att, left mssg for CB from Lynda, will wait for call
  CUST REQUESTD CHANGE IN HUNTING, FOLLOW ORDER. NO CSR FOUND. CUST
  WITH RESELLER ALEGIANCE.
A Quick AT&T-Centric Glossary

- **UNE-P RAMP**: Notes taken by customer representatives when a AT&T Consumer Local customer calls in

- **Worldnet**: Notes taken by customer representatives for AT&T WorldNet customers

- **LIFE**: AT&T Business Services Ordering System
What do I mean by “Abbreviation”?

Any word that is shortened from its normal spelling, but that should be read as if it were spelled in full.

Under this definition:

- *cust* and *mth* are abbreviations since they are clearly to be read *customer, month*

- *NATO, UN, CSR* are **not** abbreviations since they are standardly read as words ("acronyms") or sequences of letters.

- Some terms such as *LD* ("long distance") have become pretty standard, and so will not be treated as abbreviations.
### NSW Classification

#### Table I. Taxonomy of non-standard words used in hand-tagging and in the text normalization models

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Letter sequence</th>
<th>Read as word</th>
<th>Misspelling</th>
<th>Number (cardinal)</th>
<th>Number (ordinal)</th>
<th>Telephone (or part of)</th>
<th>Number as digits</th>
<th>Zip code or PO Box</th>
<th>Time</th>
<th>Date</th>
<th>Year(s)</th>
<th>Money (US or other)</th>
<th>Money tr/m/billions</th>
<th>Percentage</th>
<th>Mixed or “split”</th>
<th>Not spoken, word boundary</th>
<th>Not spoken, phrase boundary</th>
<th>Funny spelling</th>
<th>URL</th>
<th>Should be ignored</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXPN</td>
<td>LSEQ</td>
<td>ASWD</td>
<td>MSPL</td>
<td>NUM</td>
<td>NORD</td>
<td>NTFL</td>
<td>NDIG</td>
<td>NIDE</td>
<td>UADDR</td>
<td>NZIP</td>
<td>NTIME</td>
<td>NDATE</td>
<td>NYER</td>
<td>MONEY</td>
<td>BMONEY</td>
<td>PRCT</td>
<td>SLNT</td>
<td>FNNSP</td>
<td>URL</td>
<td>NONE</td>
</tr>
<tr>
<td>abbreviation</td>
<td>letter sequence</td>
<td>read as word</td>
<td>misspelling</td>
<td>number (cardinal)</td>
<td>number (ordinal)</td>
<td>telephone (or part of)</td>
<td>number as digits</td>
<td>identifier</td>
<td>number as street address</td>
<td>zip code or PO Box</td>
<td>a (compound) time</td>
<td>a (compound) date</td>
<td>money (US or other)</td>
<td>money tr/m/billions</td>
<td>percentage</td>
<td>mixed or “split”</td>
<td>not spoken, word boundary</td>
<td>not spoken, phrase boundary</td>
<td>funny spelling</td>
<td>url, pathname or email</td>
</tr>
<tr>
<td>adv, N, Y, mph, govt</td>
<td>CIA, D.C, CDs</td>
<td>CAT, proper names</td>
<td>geography</td>
<td>12, 45, 1/2, 0.6</td>
<td>May 7, 3rd, Bill Gates III</td>
<td>212 555-4523</td>
<td>Room 101</td>
<td>747, 386, 15, pc110, 3A</td>
<td>5000 Pennsylvania, 4523 Forbes</td>
<td>91020</td>
<td>3:20, 11:45</td>
<td>2/2/99, 14/03/87 (or US) 03/14/87</td>
<td>$3.45, HK$300, ¥20,000, $200K</td>
<td>$3.45 billion</td>
<td>75%, 3.4%</td>
<td>WS99, x220, 2-car</td>
<td>word boundary</td>
<td>M.bath, KENT*RLT, .really...</td>
<td>$99.9%**Whites, “…” in DECIDE…Year</td>
<td>slloooowww, sh*t</td>
</tr>
</tbody>
</table>
Percentages of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNE-P</td>
<td>18%</td>
</tr>
<tr>
<td>LIFE</td>
<td>6%</td>
</tr>
<tr>
<td>Worldnet</td>
<td>5%</td>
</tr>
</tbody>
</table>
Normalization

cci vm not wrking has not fully complted xfer to svc

Text Normalization
One Approach

Large script with lots of rules:

- s/ AN ADV / AN ADVERTISEMENT /
- s/ 2 ADVS* / TO ADVISE /
- s/ TO ADVS* / TO ADVISE /
- s/ ADVS*D* / ADVISED /
- s/ AMER[*] / AMERICA /
- s/ AMT / AMOUNT /

- Cf. U Penn Linguistic Data Consortium’s “Text Conditioning Tools”
Problem: How many ways can you spell *customer* in UNE-P RAMP?

<table>
<thead>
<tr>
<th></th>
<th>Spell</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>cmr dscnnctd</td>
<td>customer disconnected</td>
</tr>
<tr>
<td>2</td>
<td>com upset</td>
<td>customer upset</td>
</tr>
<tr>
<td>3</td>
<td>cs clg</td>
<td>customer calling</td>
</tr>
<tr>
<td>4</td>
<td>csmr clng</td>
<td>customer calling</td>
</tr>
<tr>
<td>5</td>
<td>csr called</td>
<td>customer called</td>
</tr>
<tr>
<td>6</td>
<td>cst understood</td>
<td>customer understood</td>
</tr>
<tr>
<td>7</td>
<td>cstm wnts</td>
<td>customer wants</td>
</tr>
<tr>
<td>8</td>
<td>cstmr advsd</td>
<td>customer advised</td>
</tr>
<tr>
<td>9</td>
<td>cstr claims</td>
<td>customer claims</td>
</tr>
<tr>
<td>10</td>
<td>csu req</td>
<td>customer request</td>
</tr>
<tr>
<td>11</td>
<td>csut wntd</td>
<td>customer wanted</td>
</tr>
<tr>
<td>12</td>
<td>cts called</td>
<td>customer called</td>
</tr>
<tr>
<td>13</td>
<td>cu called</td>
<td>customer called</td>
</tr>
<tr>
<td>14</td>
<td>cus advised</td>
<td>customer advised</td>
</tr>
<tr>
<td>15</td>
<td>cust care</td>
<td>customer care</td>
</tr>
<tr>
<td>16</td>
<td>custm clld</td>
<td>customer called</td>
</tr>
<tr>
<td>17</td>
<td>custo call</td>
<td>customer call</td>
</tr>
<tr>
<td>18</td>
<td>customer chngd</td>
<td>customer changed</td>
</tr>
<tr>
<td>19</td>
<td>custr upst</td>
<td>customer upset</td>
</tr>
</tbody>
</table>
Corpus-Dependent Unsupervised Abbreviation Expansion (Sproat et al. 2001)

**Problem:** given a previously unseen abbreviation, how do you use corpus-internal evidence to find the expansion into a *standard* word?

<table>
<thead>
<tr>
<th>Example:</th>
<th>cus wnt info on services and chrgs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elsewhere in Corpus:</td>
<td>. . . customer wants . . .</td>
</tr>
<tr>
<td></td>
<td>. . . wants info on vmail . .</td>
</tr>
</tbody>
</table>
A Source-Channel Language Model Approach

\[ \hat{w} \approx \arg\max_{w,t} p(o|t, w)p(t|w)p(w) \]

Where:

- **o** are the *observed text*
- **w** are the *underlying words*
- **t** are the *tags* (in this case the tags “abbreviate” and “don’t abbreviate”)
WFST-based Implementation

\[ T' = \pi_2(\text{ShortestPath}(T \circ A^{-1} \circ L)) \]

Where:

- \( T \) is text
- \( T' \) is normalized text
- \( A \) is abbreviation model
- \( L \) is language model

cf. CLG model used in ASR
Processing Steps

- Preprocess text ("splitter").

- Collect possible abbreviations and their possible expansions; use a stoplist of things not to expand.

- Train a language model on "clean" text.

- Normalize text.
Splitter

• ORD#C219XXXXXXX V2-REJ 9481 FEA DOES NOT EXIST ON ACCT/2ND ATTEMPT/TO BE PLACED IN TTID GA-CWD/IF CUS CALLS PLEASE REFER TO OM VERIBAGE

• ord # c 219XXXXXXX v 2 - rej 9481 fea does not exist on acct / 2nd attempt / to be placed in ttid ga - cwd / if cus calls please refer to om veribage

• Lextools rule-based system (also a perl version). Rules attempt to identify:
  ★ Dates, times (various formats)
  ★ telephone numbers
  ★ fractions
  ★ filenames/URL’s,
  ★ ordinals
  ★ . . .

Otherwise mixed alpha/non-alpha strings are split.
Finding Abbreviations and Potential Expansions: Dictionaries

• Large dictionary of ordinary words (320K words from U Penn XTag dictionary) augmented with 50K proper names.

Outstanding problem: abbreviations can also be words – *kit* (*kitchen*); *abt* (*about*).

• Stoplist of things to leave alone. E.g.:

  *nfcc, rcam, att, cio, asap* . . .

  (Has same problem as above)

• If a token is (almost) purely alphabetic and it’s not in the above list, treat it as a potential abbreviation

Problem: some abbreviations use non-alphabetic symbols – 2 go, 4x’s
Finding Abbreviations and Potential Expansions: Approximate Matching

- Collect bigrams of ordinary words.
- Collect token bigrams containing at least one potential abbreviation.
- Match abbreviation bigrams to word bigrams: e.g. `cus wnt → customer went`.
  Match potential abbreviation with full word if:
  - Both start with same letter
  - The abbreviation contains only letters and a few acceptable non-alphabetic symbols (', ., /)
  - No letter in the abbreviation occurs more frequently than it does in the full form
  - Letters in the abbreviation occur in (roughly) the same sequence as they do in the full form.
  So `ctsr` will match with `customer` but `clld` wouldn’t.
Finding Abbreviations and Potential Expansions: Approximate Matching

- A few “phonetic” matches are allowed: $c$ see $x$ trans-, ex-

- Some examples of matched pairs:

  - cus wnt: customer went, customer wanted, customer wants
  - bill pym: bill payment
  - insd wr: inside wire, inside wiring, inside work
  - pymnt argmnt: payment arrangement, payment agreement, payment arrangements
  - intrnt adlt: internet adult
  - line bld: line blocked, line billed

Text Normalization
Language Modeling

- Train a word trigram model with standard Katz backoff on “sanitized” text: cust business acct – trns to business office
  <ABBR> business <ABBR> <PUNC> <ABBR> to business office

- Implemented using the WFST-based LM tools that we’ve seen before
WFST-based Implementation

\[ T' = \pi_2(\text{ShortestPath}(T \circ A^{-1} \circ L)) \]
Further issues

- Run the normalization on the training data, treat the result as “truth”, and reestimate the expansions of abbreviations; can also retrain the LM on the new “truth”.

This has been shown to reduce error rates by as much as 20% on classified ads. This allows one to reestimate each component term in:

\[ p(o|t, w)p(t|w)p(w) \]

- Does limiting the detection of abbreviations to bigrams that match full word bigrams help or hurt?
Evaluation

- Separated UNE-P RAMP data into a “training” portion (779K words) and a “testing” portion (196K) words.
- Randomly selected 500 sentences (12.3K tokens) for actual testing, and hand annotated these with nominal expansions for each abbreviation (the “gold standard”). Number of abbreviations: 2177 (18%).
- Computed errors (insertions+deletions+substitutions) for each training condition on the gold standard.
- Randomly selected a subset of the cases where the output differs from the “gold standard” and decided if the automatically generated expansions were acceptable errors. Cases of acceptable errors would be different verb forms (call, called), nominalizations (adjust, adjustment) if these make sense; also lenient about things like vm versus voicemail. Used these hand annotations to estimate the percentage of errors that are true errors.
## Evaluation on UNE-P RAMP Data (12.3K tokens)

<table>
<thead>
<tr>
<th>Training Data</th>
<th>Training Data Size</th>
<th>Errors</th>
<th>Est Errors</th>
<th>Adj Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train RAMP</td>
<td>779K</td>
<td>1342</td>
<td>0.82</td>
<td>1100 (9.0%)</td>
</tr>
<tr>
<td>‡Test RAMP</td>
<td>196K</td>
<td>1380</td>
<td>0.92</td>
<td>1270 (10.0%)</td>
</tr>
<tr>
<td>‡All RAMP</td>
<td>975K</td>
<td>1254</td>
<td>0.83</td>
<td>1040 (8.5%)</td>
</tr>
<tr>
<td>LLC</td>
<td>—</td>
<td>1721</td>
<td>0.66</td>
<td>1135 (9.0%)</td>
</tr>
<tr>
<td>‡LLC+All</td>
<td>975K</td>
<td>1622</td>
<td>0.62</td>
<td>1014 (8.3%)</td>
</tr>
<tr>
<td>‡LLC+Retrain+All</td>
<td>975K</td>
<td>1624</td>
<td>0.58</td>
<td>942 (7.6%)</td>
</tr>
<tr>
<td>Worldnet</td>
<td>4.5M</td>
<td>1647</td>
<td>0.92</td>
<td>1515 (12%)</td>
</tr>
<tr>
<td>100 most freq</td>
<td>—</td>
<td>1232</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

- "LLC" is hand-crafted program
- "LLC+ALL": first run LLC then run the current normalizer on the output.
- "LLC+Retrain+ALL": run LLC on all the data, then use that data as the training data for the self-organizing approach.
- "100 most freq" simply replaces 100 most frequent abbreviations with their (most frequent) corresponding word.
Some Example Normalizations (All RAMP Model)

cst clld 2 hv cllr id blck rmvn snt local form
customer called 2 have caller id block rmvn sent local form

cst clld to verify insde wre / i cnclled his near mve on accident / cst now wnts to ploc to anther cmpny
customer called to verify inside wire / i cancelled his near move on accident / cst now wants to ploc to anther cmpny

cust no lnger wnts ld on acct
customer no longer wants ld on account

xplnd chrgs .. cust stated he w/ pay 26.45 & then w/ cancel his srvc w/ att
explained charges .. customer stated he will pay 26.45 & then will cancel his service with att
How Useful is This?

- Obviously needed if you want to read the text.
- May be needed for searching for a particular phrase (regardless of how it's spelled)
- Is it useful for text classification?
Text Classification Task

- Classify UNE-P RAMP comments into 26 different categories:

- Use BoosTexter (Schapire and Singer, 2000)

- Train on 39K examples; test on 1000