NLP Tools
LVG - Derivations (SD-Rules)

By: Dr. Chris Lu

The Lexical Systems Group

NLM. LHNCBC. CGSB

June, 2013
Table of Contents

• Introduction
  ▪ NLP Tools
    o Normalization
    o Query Expansion
  ▪ SD-Rules
    ▪ Derivations in the Lexical Tools
    ▪ Systematic Approach
    ▪ SD-Rules Set Optimization
  ▪ Results
• Questions
Introduction - NLP

• Natural Language (English)
  ▪ is ordinary language that humans use naturally
  ▪ may be spoken, signed, or written

• Natural Language Processing
  ▪ NLP is to process human language to make their information accessible to computer applications
  ▪ The goal is to design and build software that will analyze, understand, and generate human language
  ▪ Most NLP applications require knowledge from linguistics, computer science, and statistics
NLP Example

- Information retrieval
- Filter
- Summarize
- Alert & suggestion
- Questions answering
- …
NLP System

Domain Knowledge

- EMR (Electronic Medical Records)
- MEDLINE Article/Abstract
- ...

NLP algorithm programs

Features:
- Information retrieval
- Filter
- Summarize
- Alert & suggestion
- Questions answering
- ...

Structured Data

- The SPECIALIST Lexicon
- UMLS semantic network
- ...

• Programs
- The SPECIALIST Lexicon
- UMLS semantic network
- …
NLP – Concepts

• UMLS (Unified Medical Language System)
  ▪ is a comprehensive thesaurus and ontology of biomedical concepts
  ▪ created in 1986 by NLM
  ▪ provides terms to concepts mapping from different controlled vocabularies sources, such as ICD-10, MeSH, SNOMED CT, etc.
  ▪ includes:
    o Metathesaurus
    o Semantic Network
    o SPECIALIST Lexicon and Lexical Tools
Challenge in Concept Mapping

• Terms have multiple concepts
  ▪ Example: cold (7 CUIs in UMLS-2013AA)
    o Cold Temperature
    o Common Cold
    o Cold Therapy
    o Cold Sensation
    o etc..
  ➢ Word Sense Disambiguation (WSD)

• Concepts has variety of ways to express
  ▪ Example: Hodgkin's Disease
  ➢ Normalization
  ➢ Query Expansion
NLP - Norm

- Hodgkin Disease
- HODGKINS DISEASE
- Hodgkin's Disease
- Disease, Hodgkin's
- HODGKIN'S DISEASE
- Hodgkin's disease
- Hodgkins Disease
- Hodgkin's disease NOS
- Hodgkin's disease, NOS
- Disease, Hodgkins
- Diseases, Hodgkins
- Hodgkins Diseases
- Hodgkins disease
- hodgkin's disease
- Disease;Hodgkins
- Disease, Hodgkin
- ...

Terms in Corpus

normalize

Index

Indexed Database Normalized String
Hodgkin’s Disease

Query

norm

Normed term

disease hodgkin

Indexed Database
Normalized String

Results that matches the normalized query

SQL
NLP – Query Expansion

Indexed Database Normalized String

None

C0206504
Tympanic Membrane Perforation
Lexical Variants

- To increase recall & precision

<table>
<thead>
<tr>
<th></th>
<th>Query expansion (Recall)</th>
<th>POS Tagging (Precision)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>perforated ear drum</td>
<td>saw</td>
</tr>
</tbody>
</table>
| UMLS-CUI         | None                    | • C1947903|See  
                  |                         | • C0183089|saw (device) |
| Process          | perforation ear drum    | noun                   |
| UMLS-CUI         | C0206504                | • C0183089             |
| Preferred term   | Tympanic Membrane       | saw (device)           |
|                  | Perforation             |                        |
**NLP Core Tasks**

Example: Information retrieval (search engine)
- **Tokenize & tagging (entity recognition)**
  - break inputs into words <Text Tools, wordInd>
  - POS tagging <dTagger>
  - Other annotation <Visual Tagging Tool, VTT>
- spelling check
  - suggest correct spelling for misspelled words <gSpell>
- lexical variants (normalization/query expansion)
  - spelling variants, inflectional/uninflectional variants, synonyms, acronyms/abbreviations, expansions, derivational variants, etc. <Lexical Tools, LexAccess, LexCheck, STMT>
- semantic knowledge (concept mapping)
  - map text to Metathesaurus concepts <MetaMap, MMTX, STMT>
  - Word Sense Disambiguation <TC – StWSD>
NLP Tools by LSG

The SPECIALIST NLP Tools

LexBuild

The SPECIALIST LEXICON

Lexical Tools

LVG-Derivations (SD-Rules)

Text Tools

NLP Applications

LexCheck

LexAccess


Derivational Variants

• Words are related by a derivational process
  ▪ Used to create new words based on existing words
  ▪ Meaning change (related)
  ▪ Category may change
  ▪ Derivational process: suffix, prefix, and conversion

• Focus on relatedness (no direction)
Derivation Types (-kdt)

• Example (kind|adj):
  ▪ zeroD: kind|adj|kind|noun
  ▪ prefixD: kind|adj|unkind|adj
  ▪ suffixD: kind|adj|kindly|adv
Derivational Pair

• Each link and the associated two nodes in derivational network define a derivational pair
• Includes base forms and syntactic category information
• Bi-directional
• Only involves one or none derivational affix
• Lvg format: base 1|category 1|base 2|category 2
• Examples:
  ▪ kind|adj|kindness|noun
  ▪ kind|adj|kindly|adv
  ▪ kind|adj|unkind|adj
  ▪ kind|adj|kind|noun
Derivations in LVG

• 7 flow components (62):
  ▪ -f:d
  ▪ -f:dc
  ▪ -f:R
  ▪ -f:G
  ▪ -f:Ge
  ▪ -f:Gn
  ▪ -f:v

• 3 flow specific options (39):
  ▪ -kd: 1|2|3 (default: 1)
  ▪ -kdn: B|N|O (default: O)
  ▪ -kdt: Z|S|P (default: ZSP)
• `shell> lvg -f:d -p -SC -SI

  - Please input a term (type "Ctl-d" to quit) >

  hyperuricemic

  hyperuricemic|hyperuricemic|<noun>|<base>|d|1|
  hyperuricemic|hyperuricemia|<noun>|<base>|d|1|
  hyperuricemic|hyperuricemic|<adj>|<base>|d|1|
## Derivations Generation

- Before 2011-, issues of precision and recall
- A new systematic approach to automatically generating derivational variants using LVG conjunction with Specialist Lexicon:

<table>
<thead>
<tr>
<th>2012</th>
<th>2013</th>
<th>2014</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefixD &amp; zeroD</td>
<td>suffixD</td>
<td>SD-Rules</td>
</tr>
</tbody>
</table>

### References:
Systematic Approach

• Better coverage:
  ▪ Facts: cover all dPairs known to Lexicon (grow proportionally with Lexicon annually)

<table>
<thead>
<tr>
<th></th>
<th>2011-</th>
<th>2012</th>
<th>2013</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>4,559</td>
<td>89,950</td>
<td>121,078</td>
</tr>
</tbody>
</table>

• Better precision:
  ▪ Mainly relies on facts: virtually 100% accurate

• Derivations not in Lexicon?
Derivational Flow

• Facts
  ▪ derivational pairs database table

<table>
<thead>
<tr>
<th>Base-1</th>
<th>Cat-1</th>
<th>EUI-1</th>
<th>Base-2</th>
<th>Cat-2</th>
<th>EUI-2</th>
<th>Negation</th>
<th>Type</th>
<th>prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>care</td>
<td>noun</td>
<td>E0015334</td>
<td>precare</td>
<td>noun</td>
<td>E0611704</td>
<td>O</td>
<td>P</td>
<td>pre</td>
</tr>
<tr>
<td>care</td>
<td>noun</td>
<td>E0015334</td>
<td>careless</td>
<td>adj</td>
<td>E0015344</td>
<td>N</td>
<td>S</td>
<td>None</td>
</tr>
<tr>
<td>care</td>
<td>noun</td>
<td>E0015334</td>
<td>care</td>
<td>verb</td>
<td>E0015335</td>
<td>O</td>
<td>Z</td>
<td>None</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

• SD-Rules
  ▪ Use exceptions to increase precision

EXAMPLE: retirement| noun | retire | verb
RULE: ment$ | noun | $ | verb
EXCEPTION: apartment | apart;
EXCEPTION: basement | base;
EXCEPTION: department | depart;
...
SD-Rules (Trie)

- retirement|noun => retire|verb

EXAMPLE: retire|verb|retirement|noun
RULE: $|verb|ment$|noun
EXCEPTION: apart|apartment;
...

EXAMPLE: conformant|adj|conformance|noun
RULE: ance$|noun|ant$|adj
EXCEPTION: ambulant|ambulance;
...

EXAMPLE: relaxant|adj|relax|verb
RULE: ant$|adj|$|verb
EXCEPTION: important|import;
...

EXAMPLE: conformant|adj|conformance|noun
RULE: ance$|noun|ant$|adj
EXCEPTION: ambulant|ambulance;
...

EXAMPLE: fluent|adj|fluency|noun
RULE: ency$|noun|ent$|adj
EXCEPTION: emergency|emergent;
...
Facts Generation

Lexicon

Automatic generation:
• zeroD
• prefixD
• suffixD

Raw dPairs

Automatic tagging

Experts’ tagging
- type
- negation
- etc.

Derivation table
# SD-Fact Data

- Original SD generating rules in SD-Facts process:

<table>
<thead>
<tr>
<th>No.</th>
<th>Rules to generate Raw SD-Pairs</th>
<th>Raw Retrieved</th>
<th>Valid Relevant</th>
<th>Invalid Irrelevant</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$</td>
<td>adj</td>
<td>ness$</td>
<td>noun</td>
</tr>
<tr>
<td>2</td>
<td>ability$</td>
<td>noun</td>
<td>able$</td>
<td>adj</td>
</tr>
<tr>
<td>3</td>
<td>ization$</td>
<td>noun</td>
<td>ize$</td>
<td>verb</td>
</tr>
<tr>
<td>4</td>
<td>osis$</td>
<td>noun</td>
<td>otic$</td>
<td>366</td>
</tr>
<tr>
<td>5</td>
<td>le$</td>
<td>adj</td>
<td>ly$</td>
<td>adv</td>
</tr>
<tr>
<td>71</td>
<td>ious$</td>
<td>adj</td>
<td>ly$</td>
<td>noun</td>
</tr>
<tr>
<td>72</td>
<td>ant$</td>
<td>adj</td>
<td>ate$</td>
<td>verb</td>
</tr>
<tr>
<td>73</td>
<td>$</td>
<td>noun</td>
<td>ist$</td>
<td>noun</td>
</tr>
<tr>
<td>93</td>
<td>ia$</td>
<td>noun</td>
<td>ian$</td>
<td>noun</td>
</tr>
<tr>
<td>94</td>
<td>a$</td>
<td>noun</td>
<td>an$</td>
<td>noun</td>
</tr>
<tr>
<td>95</td>
<td>gram$</td>
<td>noun</td>
<td>graphy$</td>
<td>noun</td>
</tr>
<tr>
<td>96</td>
<td>gram$</td>
<td>noun</td>
<td>graphic$</td>
<td>adj</td>
</tr>
<tr>
<td>97</td>
<td>$</td>
<td>verb</td>
<td>ably$</td>
<td>adv</td>
</tr>
</tbody>
</table>
SD-Rules Optimization

• Objective:
  To find an optimized set of SD-Rules to reach best performance (precision and recall)
  ▪ to have high precision (95%)
  ▪ to cover more derivations (recall) that are not in Lexicon

• Assumption:
  Use Lexicon as the testing corpus by assuming Lexicon is a representable subset of general English
Step 1 - Normalize

• Remove duplicates
  ▪ Unify bi-directional SD-Rules (alphabetic order sorting)

• Remove overlap (child rules)
  ▪ Example:
    magic|noun|E0038555|magical|adj|E0038557
    ➢ $|noun|al$|adj|2013|ORG_RULE|PARENT
    ➢ ic$|noun|ical$|adj|2013|ORG_RULE|CHILD

• Normalize 97 to 87 SD-Rules
## Step 1 - Normalize

- **Remove Child-Rules:**

<table>
<thead>
<tr>
<th>Parent-rules (9)</th>
<th>Child-rules (10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>adj</td>
</tr>
<tr>
<td>$</td>
<td>noun</td>
</tr>
<tr>
<td>a$</td>
<td>noun</td>
</tr>
<tr>
<td>a$</td>
<td>noun</td>
</tr>
<tr>
<td>a$</td>
<td>noun</td>
</tr>
<tr>
<td>ance$</td>
<td>noun</td>
</tr>
<tr>
<td>ation$</td>
<td>noun</td>
</tr>
<tr>
<td>ency$</td>
<td>noun</td>
</tr>
<tr>
<td>sis$</td>
<td>noun</td>
</tr>
<tr>
<td>osis$</td>
<td>noun</td>
</tr>
</tbody>
</table>
Step 2 – Performance

• A good SD-Rule: has high precision and high frequency

• A good set of SD-Rules: includes better SD-Rules to reach better system performance for:
  ▪ higher system precision (> 95%)
  ▪ higher system recall
  ▪ more SD-Rules (for better coverage)
Step 2 – System Performance

- Sort all SD-Rules by:
  - precision (valid No. / raw No.)
  - raw No. (frequency).
  - alphabetic order of SD-Rules

- System performance:
  - System precision (cumulative):
    \[ P = \frac{\text{relevant, retrieved}}{\text{retrieved}} \]
  - System recall:
    \[ R = \frac{\text{relevant, retrieved}}{\text{relevant}} \]

  - More SD-Rules (for tie-breaker)
Step 3 – Optimization

• The optimal set has the best system performance

• Parent-Child SD-Rules
  Compare system performance of Parents (9) to Child SD-Rules (10)

• Add New SD-Rules:
  ▪ from nomD
  ▪ from original Facts
  ▪ form suggestions
### Step 3.1 – Optimization

- Evaluate Parent-Child-Grandchild Rules:
  - Only case 2 provides better results while replacing parent rule by child rule
  - Case 2.3 has the best results among case 2

<table>
<thead>
<tr>
<th>ID</th>
<th>Parent-Rule</th>
<th>Candidate Child-Rules</th>
<th>Rule No.</th>
<th>Precision</th>
<th>Cutoff SD-Rule</th>
<th>Sys P</th>
<th>Sys R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Parent-rule only (Baseline)</td>
<td>No child-Rule</td>
<td>60</td>
<td>73.68%</td>
<td>a$</td>
<td>noun</td>
<td>iasis$</td>
</tr>
<tr>
<td>1.1</td>
<td>$</td>
<td>adj</td>
<td>lity$</td>
<td>noun</td>
<td>c$</td>
<td>adj</td>
<td>city$</td>
</tr>
<tr>
<td>1.2</td>
<td>$</td>
<td>adj</td>
<td>lity$</td>
<td>noun</td>
<td>ic$</td>
<td>adj</td>
<td>licity$</td>
</tr>
<tr>
<td>2.1</td>
<td>$</td>
<td>noun</td>
<td>al$</td>
<td>adj</td>
<td>n$</td>
<td>noun</td>
<td>nal$</td>
</tr>
<tr>
<td>2.2</td>
<td>$</td>
<td>noun</td>
<td>al$</td>
<td>adj</td>
<td>on$</td>
<td>noun</td>
<td>onal$</td>
</tr>
<tr>
<td>2.3</td>
<td>$</td>
<td>noun</td>
<td>al$</td>
<td>adj</td>
<td>ion$</td>
<td>noun</td>
<td>ional$</td>
</tr>
<tr>
<td>2.4</td>
<td>$</td>
<td>noun</td>
<td>al$</td>
<td>adj</td>
<td>tion$</td>
<td>noun</td>
<td>tional$</td>
</tr>
<tr>
<td>3.1</td>
<td>a$</td>
<td>noun</td>
<td>an$</td>
<td>adj</td>
<td>No candidate child-rule found</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>8.1</td>
<td>ency$</td>
<td>noun</td>
<td>ent$</td>
<td>adj</td>
<td>No candidate child-rule found</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.1</td>
<td>sis$</td>
<td>noun</td>
<td>tic$</td>
<td>adj</td>
<td>e$</td>
<td>noun</td>
<td>etic$</td>
</tr>
</tbody>
</table>
3.1 Optimization Example

- Sorted SD-Rules of case 2.3:

<table>
<thead>
<tr>
<th>No.</th>
<th>Rule Precision</th>
<th>Raw Retrieved</th>
<th>Valid Relevant</th>
<th>Invalid Relevant</th>
<th>SD-Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100.00%</td>
<td>2723</td>
<td>2723</td>
<td>0</td>
<td>$</td>
</tr>
<tr>
<td>2</td>
<td>100.00%</td>
<td>1278</td>
<td>1278</td>
<td>0</td>
<td>ability$</td>
</tr>
<tr>
<td>3</td>
<td>100.00%</td>
<td>326</td>
<td>326</td>
<td>0</td>
<td>le$</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>64</td>
<td>62.65%</td>
<td>332</td>
<td>208</td>
<td>124$</td>
<td>noun</td>
</tr>
<tr>
<td>65</td>
<td>60.66%</td>
<td>183</td>
<td>111</td>
<td>72 ar$</td>
<td>adj</td>
</tr>
<tr>
<td>66</td>
<td>58.08%</td>
<td>582</td>
<td>338</td>
<td>244 al$</td>
<td>adj</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>84</td>
<td>0.37%</td>
<td>273</td>
<td>1</td>
<td>272 a$</td>
<td>noun</td>
</tr>
<tr>
<td>85</td>
<td>0.00%</td>
<td>358</td>
<td>0</td>
<td>358 gram$</td>
<td>noun</td>
</tr>
<tr>
<td>86</td>
<td>0.00%</td>
<td>228</td>
<td>0</td>
<td>228 gram$</td>
<td>noun</td>
</tr>
<tr>
<td>87</td>
<td>0.00%</td>
<td>57</td>
<td>0</td>
<td>57 $</td>
<td>verb</td>
</tr>
</tbody>
</table>
### 3.1 Optimization Example

- Sorted SD-Rules of case 2.3:

<table>
<thead>
<tr>
<th>No.</th>
<th>Rule Precision</th>
<th>Raw</th>
<th>Valid</th>
<th>InV.</th>
<th>SD-Rule</th>
<th>Accum Total</th>
<th>Accum Valid</th>
<th>System Precision</th>
<th>System Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100.00%</td>
<td>2723</td>
<td>2723</td>
<td>0</td>
<td>$</td>
<td>adj</td>
<td>ness$</td>
<td>noun</td>
<td>2723</td>
</tr>
<tr>
<td>2</td>
<td>100.00%</td>
<td>1278</td>
<td>1278</td>
<td>0</td>
<td>ability$</td>
<td>noun</td>
<td>able$</td>
<td>adj</td>
<td>4001</td>
</tr>
<tr>
<td>3</td>
<td>100.00%</td>
<td>326</td>
<td>326</td>
<td>0</td>
<td>le$</td>
<td>adj</td>
<td></td>
<td>ly$</td>
<td>adv</td>
</tr>
</tbody>
</table>

... ... ... ... ...

| 64  | 62.65%         | 332 | 208   | 124  | $|noun|ist$|noun    | 36673       | 34907       | 95.18%          | 94.00%       |
| 65  | 60.66%         | 183 | 111   | 72   | ar$|adj||e$|noun   | 36858       | 35018       | **95.01%**      | 94.30%       |
| 66  | 58.08%         | 582 | 338   | 244  | al$|adj||e$|noun   | 37438       | 35356       | 94.44%          | 95.21%       |

... ... ... ... ...

| 84  | 0.37%          | 273 | 1272  | 1    | a$|noun|an$|noun  | 42732       | 36673       | 87.33%          | 98.75%       |
| 85  | 0.00%          | 358 | 358   | 0    | gram$|noun|graphy$|noun     | 43090       | 36673       | 86.60%          | 98.75%       |
| 86  | 0.00%          | 228 | 228   | 0    | gram$|noun|graphic$|adj      | 43318       | 36673       | 86.13%          | 98.75%       |
| 87  | 0.00%          | 57  | 57    | 0    | $|verb|ably$|adv    | 43375       | 36673       | 86.02%          | 98.75%       |

(37136)
3.1 Optimization Results

- Best set includes 65 rules with S.P. of 95.01% and S.R. of 94.30%

Set 2.3 (optimization of normalized set, 65/87 rules)
3.2 Enhancement – Add More Rules

- Use same method to evaluate/add new SD-Rules
  - From nomD (4), 1 is the parent rule
  - From factD (5)
  - From others’ suggestions (1)
- Final best set includes 73 rules with S.P. of 95.30% and S.R. of 95.01%
Result - Noise Reduction

- Smoothing algorithm – simple moving average of 3, 5, 7 window size
- The intersections are all around 95% for all cases
- Confirm our optimized goal of 95% S.A. is a good choice
Optimization Summary

- Sort by:
  - precision (= valid No. / raw No.)
  - raw No. (frequency)
  - alphabetic order of SD-Rules (remove duplications)
- Find performance:
  - precision (cumulative): above 95%
  - recall: coverage
- Evaluate related Parent-Child Rules:
  - remove all child-rules
  - decompose parent-rules
  - evaluated performance (precision and recall)
- Get the best set of SD-Rules (with best performance - intersection of curves of precision and recall)
Process Summary

1. A new SD-Rule
   - Get tagging stats data

2. SD-Rules Set

3. Remove Duplicated Rules

4. Evaluate Parent-Child Rules:
   - Remove all child-rules
   - Decompose parent-rule
   - Compare system performance between parent-child rules

5. Find the system performance

6. Find the optimized set of SD-Rules
Results

• A comprehensive derivational features in Lexical Tools:
  ▪ Type options: prefixD, suffixD, zeroD
  ▪ Negation options
• A maintainable and scalable system for generating derivations with the Lexicon’s annual release

• Better precision:
  ▪ in Lexicon: virtual 100%
  ▪ not in Lexicon (SD-Rules set): above 95.30%
• Better recall:
  ▪ in Lexicon: 100% for the candidate SD-Rules
  ▪ not in Lexicon (SD-Rules set): about 95.01%
Future Work

• Annual routine update with lexicon release

• Enhancement:
  ▪ prefixD: work on more prefixes (2014)
  ▪ suffixD: work on more candidate SD-Rules (2014)

• Analysis in prefixD, suffixD and zeroD

• Assumption (from Lexicon to English):
  ▪ Is Lexicon a representable subset of general English?
Questions