Term Based Tools

- Text-to-Concept Mapping Tools
- Indexing Tools
- Text Categorization

Corpus and Document Based Tools

- Lexical Tools
- Term Lookup

Sentence/Phrase/Term/Word Tokenizers
- POS Tagging
- Spelling Suggestion

SPECIALIST Lexicon
SPECIALIST.nlm.nih.gov

About

Projects

Documents

The Lexicon

Lexicon

Term Lookup

Term Manipulation Tools

Spelling Suggestion

Document Tokenization Tools

Text Categorization Tool

POS Tagger

MetaMap Transfer (MMTx)
mmtx.nlm.nih.gov
Lexicon
SPECIALIST Lexical Tools

- Word Indexing
- Variant Generation
- Term Normalization

Term Based Tools
Uses

- Term Transformation
- Query Expansion
- Term normalization
- Building indexes
- Component of controlled vocabulary building tools
- Syntactic parsing
- Component of search engines
- Component of text-to-concept mapping tools
- Component of automated document indexing tools
- Component of text summarization tools
- Component of data-mining tools
Lexical Variant Generation

- inflections
  - treats
  - treating
  - treated

- nominalizations
  - treatment
  - treatments

- combinations
  - treatability

- derivations
  - treatable

- Term Based Tools

- treat
  - treats
  - treating
  - treated
  - treatment
  - treatments
  - treatable
  - treaty
  - treater
Lexical Variant Generation

- colour
- coloring
- colored
- colors
- inflections
- nominalizations
- combinations
- derivations
- cololessness
- Chromaticness
- Chromaticities
- Spelling variants

- synonyms:
  - chromatic
  - colorant
  - colorful
  - colorless
Lexical Variant Generation

secondarily

Lexical Variant Generation

second

inflections

nominalizations

derivations

synonyms

acronyms

seconds

s

s’s

serous

Ser

SOR

secant

secondarily

secondly

secondary

Term Based Tools

Acronym expansions

Acronym expansions
The tools can be arranged so that the output of one is the input to another.
Term Normalization

**Norm** abstracts away from:
- case
- punctuation
- word order
- stop words
- possessive forms
- inflectional variation
- spelling variation
- normalizes diacritics/ligatures/symbols
Term Normalization Examples

• Word order
  – Upper left lobe of lung
  – Left upper lobe of lung

• Possessive Forms
  – Grave’s Disease
  – Graves Disease
  – Graves’ Disease

• Diacritic/Ligature/Symbol Normalization
  – entrée, anæsthesia, β-blockers, Medline®
## Term Normalization Example

<table>
<thead>
<tr>
<th>Term</th>
<th>Normalized Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hodgkin Disease</td>
<td>Hodgkin's disease, NOS</td>
</tr>
<tr>
<td>HODGKINS DISEASE</td>
<td>Disease, Hodgkins</td>
</tr>
<tr>
<td>Hodgkin's Disease</td>
<td>Diseases, Hodgkins</td>
</tr>
<tr>
<td>Disease, Hodgkin's</td>
<td>Hodgkins Diseases</td>
</tr>
<tr>
<td>HODGKIN'S DISEASE</td>
<td>Hodgkins disease</td>
</tr>
<tr>
<td>Hodgkin's disease</td>
<td>hodgkin's disease</td>
</tr>
<tr>
<td>Hodgkins Disease</td>
<td>Disease; Hodgkins</td>
</tr>
<tr>
<td>Hodgkin's disease NOS</td>
<td>Disease, Hodgkin</td>
</tr>
</tbody>
</table>

Note: A normalized form is not necessarily itself a readable term. It is a hash.

[Normalization web tool]
Lgt, a GUI Example
Command Line Example

```
> lvg -f:i -SC -SI
leave
leave|leave|<noun>|<base>|i|1|
leave|leave|<noun>|<singular>|i|
leave|leaves|<noun>|<plural>|i|1|
leave|left|<verb>|<past>|i|1|
leave|left|<verb>|<pastPart>|i|1|
leave|leave|<verb>|<base>|i|1|
leave|leave|<verb>|<pres1p23p>| |
leave|leave|<verb>|<infinitive>|i|
leave|leaves|<verb>|<pres3s>|i|1
leave|leaving|<verb>|<presPart>|
```

```
> lvg -f:i
leave
leave|leave|128|1|i|1|
leave|leave|128|512|i|1|
leave|leaves|128|8|i|1|
leave|left|1024|64|i|1|
leave|left|1024|32|i|1|
leave|leave|1024|1|i|1|
leave|leave|1024|262144|
leave|leave|1024|1024|i|1|
leave|leaves|1024|128|i|1|
leave|leaving|1024|16|i|1|
```
Command Line Example: Output Fields Explained

> lvg -f:i
leave

<table>
<thead>
<tr>
<th>Input Term</th>
<th>Output Term</th>
<th>Categories</th>
<th>Inflections</th>
<th>Flow history</th>
<th>Flow Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>leave</td>
<td>leave</td>
<td>128</td>
<td>1</td>
<td>i</td>
<td>1</td>
</tr>
</tbody>
</table>
Outline of the needed pieces:

- import gov.nih.nlm.nls.lvg.Api.*;

- NormApi api = new NormApi();

- Vector<String> out = api.Mutate(inStr);

- api.CleanUp();
import java.util.*;
import gov.nih.nlm.nls.lvg.Api.*;

public class simplestApi
{
    public static void main(String[ ] args)
    {
        NormApi api = new NormApi( ); // instantiate a NormApi object

        try // Process
        {
            Vector<String> out = api.Mutate("inputs");
            for(int i = 0; i < out.size( ); i++) // print out result
            {
                System.out.println(out.elementAt(i));
            }
        }
        catch (Exception e) {
        }
        api.CleanUp(); // make sure to clean up
    }
}
Norm API Example (2)

• To compile and run:

CLASSPATH = ${CLASSPATH}:
${LVG_DIR}:
${LVG_DIR}/lib/lvg2007dist.jar:
Application

Metathesaurus English Strings

norm

Normalized string index MRXNS.ENG

WordInd

Normalized word index MRXNW.ENG
Application

Query → norm → Normed term → Normalized string index → Normalized word index → SUIS → Metathesaurus concepts that match the normalized query
Example

Dry Eyes Syndrome

Query

norm

Normed term
dry eye syndrome
Normed term

ENG dry eye syndrome C0013238L0013238S0004019
ENG dry eye syndrome C0013238L0013238S0035652
ENG dry eye syndrome C0013238L0013238S0090228
ENG dry eye syndrome C0013238L0013238S0090454
ENG dry eye syndrome C0013238L0013238S0220550
ENG dry eye syndrome C0013238L0013238S0368350
ENG dry eye syndrome C0013238L0013238S1459074
Example (Cont.)

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0013238</td>
<td>Dry Eye Syndromes</td>
<td>S0035652</td>
<td>Dry Eye Syndromes</td>
</tr>
<tr>
<td>C0013238</td>
<td>Dry eye syndrome</td>
<td>S0004019</td>
<td>Dry eye syndrome</td>
</tr>
<tr>
<td>C0013238</td>
<td>Dry Eye Syndrome</td>
<td>S0368350</td>
<td>Dry Eye Syndrome</td>
</tr>
<tr>
<td>C0013238</td>
<td>dry eye syndrome</td>
<td>S1459074</td>
<td>dry eye syndrome</td>
</tr>
<tr>
<td>C0013238</td>
<td>Syndrome, Dry Eye</td>
<td>S0090228</td>
<td>Syndrome, Dry Eye</td>
</tr>
<tr>
<td>C0013238</td>
<td>Dry, eye syndrome</td>
<td>S0220550</td>
<td>Dry, eye syndrome</td>
</tr>
<tr>
<td>C0013238</td>
<td>Syndromes, Dry Eye</td>
<td>S0090454</td>
<td>Syndromes, Dry Eye</td>
</tr>
</tbody>
</table>
Introduction

The Tools

- GSpell
- Usage
- Output
- API

Bag-O-WordsPlus

Usage
API
Spelling Retrieval Tools

• **GSpell**
  – A term retrieval tool
  – N-gram nearest neighbor algorithm
  – MetaPhone phonetic spelling normalization
  – Homophones
  – Common misspellings
  – Candidates sorted by an edit distance and frequency of occurrence from a corpus

• **BagOWordsPlus**
  – a phrase retrieval tool
  – uses correctly spelled words within the phrase to limit possible candidates
  – uses GSpell only when it has to.
GSpell: Usage

Usage

**GSpellFind.** `[sh|bat]`

```
--dictionary=NameOfDictionary
[--inputFile=Source] [--outputFile=target]
[--truncate=N] [--considerNCandidates=N]
[--maxEditDistance=N]
```

**GSpellIndex.** `[sh|bat]`

```
--dictionary=NameOfDictionary
--inputFile=SourceFile
[--reportTime] [--version] [--help]
```
<table>
<thead>
<tr>
<th>Input Term</th>
<th>Suggestion</th>
<th>Edit Distance</th>
<th>Rank</th>
<th>Method</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>anomonomous</td>
<td><strong>anonymous</strong></td>
<td>1.0</td>
<td>0.87</td>
<td>NGrams</td>
<td></td>
</tr>
<tr>
<td>anomonomous</td>
<td><strong>allonomous</strong></td>
<td>2.0</td>
<td>0.58</td>
<td>NGrams</td>
<td></td>
</tr>
<tr>
<td>anomonomous</td>
<td><strong>autonomous</strong></td>
<td>2.0</td>
<td>0.58</td>
<td>NGrams</td>
<td></td>
</tr>
<tr>
<td>anomonomous</td>
<td><strong>anadromous</strong></td>
<td>3.0</td>
<td>0.29</td>
<td>NGrams</td>
<td></td>
</tr>
<tr>
<td>anomonomous</td>
<td><strong>analogous</strong></td>
<td>3.0</td>
<td>0.29</td>
<td>NGrams</td>
<td></td>
</tr>
<tr>
<td>anomonomous</td>
<td><strong>anomalous</strong></td>
<td>3.0</td>
<td>0.29</td>
<td>NGrams</td>
<td></td>
</tr>
<tr>
<td>anomonomous</td>
<td><strong>anonymously</strong></td>
<td>3.0</td>
<td>0.29</td>
<td>NGrams</td>
<td></td>
</tr>
<tr>
<td>anomonomous</td>
<td><strong>anonymes</strong></td>
<td>3.0</td>
<td>0.29</td>
<td>Metaphone</td>
<td></td>
</tr>
<tr>
<td>anomonomous</td>
<td><strong>anonyms</strong></td>
<td>3.0</td>
<td>0.29</td>
<td>Metaphone</td>
<td></td>
</tr>
<tr>
<td>anomonomous</td>
<td><strong>acoprous</strong></td>
<td>4.0</td>
<td>0.11</td>
<td>NGrams</td>
<td></td>
</tr>
</tbody>
</table>
import gov.nih.nlm.nls.gspell.GSpell;  // <-------These come from the gspell.jar
import gov.nih.nlm.nls.gspell.Candidate;

GSpell gspell = new GSpell( _dictionaryName,
                           GSpell.READ_ONLY );

Vector candidates = gspell.find( aTerm );
if ( candidates != null )
  for ( int i = 0; i < candidates.length; i++ )
    System.out.println(candidates[i].toString());
else
  System.out.println("No Suggestions");

gspell.cleanup();
SPECIALIST Text Tools

- LexicalLookup
- NpParser
- VariantLookup
- POS tagger
- Document Indexing
Abstract: The in-vitro antioxidant activity of natural (essential oils, vitamin E) or synthetic substances (tert-butyl hydroxy anisole (BHA), Trolox) has been evaluated by monitoring volatile carbonyl compounds released in model lipid systems subjected to peroxidation. The procedure employed methodology previously developed for the determination of carbonyl compounds as their pentafluorophenylhydrazine derivatives which were quantified, with high sensitivity, by means of capillary gas chromatography with electron capture detection. Linoleic acid and sunflower oil were used as model lipid systems. Lipid peroxidation was induced in linoleic acid by the Fe$_2^+$ ion (1 mmol L$^{-1}$, 37°C, 12 h) and in sunflower oil by heating in the presence of O$_2$ (220°C, 2 h).
Specialist TextTools

Container Classes: Entity Diagram
gov.nih.nlm.nls.nlp.textfeatures

Document
Vector getSections()
Vector getSentences()
Vector getPhrases()
Vector getLexicalElem
Vector getTokens()

Section
Vector getSentences()
Vector getPhrases()
Vector getLexicalElem
Vector getTokens()

Sentence
Vector getPhrases()
Vector getLexicalElem
Vector getTokens()

Lexical Element
getTokens()

Phrase
Vector getLexicalElemements()
Vector getTokens()

Token
These are words

Contains many relationship

These are terms
## Specialist TextTools

**Container Classes: Entity Diagram, more details**

gov.nih.nlm.nls.nlp.textfeatures

### Collection

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection()</td>
</tr>
<tr>
<td>Collection(GlobalBehavior pSettings)</td>
</tr>
<tr>
<td>Collection(StringBuffer pText)</td>
</tr>
<tr>
<td>Collection(String pFileName,</td>
</tr>
<tr>
<td>GlobalBehavior pSettings)</td>
</tr>
<tr>
<td>Vector getDocuments()</td>
</tr>
</tbody>
</table>

### Variant

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>String getTerm()</td>
</tr>
<tr>
<td>Vector getTokens()</td>
</tr>
<tr>
<td>int getCategories()</td>
</tr>
<tr>
<td>int getDistance()</td>
</tr>
<tr>
<td>int getHistory()</td>
</tr>
<tr>
<td>String getNormalizedTerm()</td>
</tr>
<tr>
<td>int getOrigCat()</td>
</tr>
<tr>
<td>String getOrigTerm()</td>
</tr>
<tr>
<td>LexicalElement getParent()</td>
</tr>
</tbody>
</table>

### Document (continued)

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document()</td>
</tr>
<tr>
<td>Document( File pFile)</td>
</tr>
<tr>
<td>Document(GlobalBehavior pSettings)</td>
</tr>
<tr>
<td>Document(String pFileName)</td>
</tr>
</tbody>
</table>
**LexicalLookup** segments text into

- Sections
- Sentences
- **Terms**
- Lexical Entries
- Words
Segments text into

- Sections
- Sentences
- **Noun Phrases**
- Terms
- Words
- Lexical Entries
The knowledge and expectations of parents about the role of antibiotic treatment in upper respiratory tract infection - a survey among parents attending the primary physician with their sick child.
### SPECIALIST TextTools

#### Classes

**LexicalLookupAPI**

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>LexicalLookupAPI( gov.nih.nlm.nls.utils.GlobalBehavior pSettings )</td>
</tr>
<tr>
<td>LexicalLookupAPI( String[] args)</td>
</tr>
</tbody>
</table>

**Parse**

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse( String[] args)</td>
</tr>
</tbody>
</table>
### Method Summary

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>processCollection(Collection pCollection )</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>processDocument(Document pDocument )</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>processSentence(Sentence pSentence )</td>
<td></td>
</tr>
<tr>
<td>Sentence</td>
<td>processSentence(String pString )</td>
<td></td>
</tr>
</tbody>
</table>
**Class GlobalBehavior: Constructor Summary**

<table>
<thead>
<tr>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlobalBehavior(String pName, String registryFile)</td>
</tr>
<tr>
<td>GlobalBehavior(String pName, String registryFile, String[] args)</td>
</tr>
</tbody>
</table>

**Properties File Location:**

`/somepath/textTools_v0.X.X/nls/nlp/config/NLPRegistry.cfg`

**Java Classpath requirement:**

`/somepath/textTools_v0.X.X/nls/nlp/config/`

**NLPRegistry.cfg: Example Contents:**

`-015|--annotationFormat1|boolean|false|Simple Annotation format`

**Command line arguments: Example Contents:**

`--annotationFormat1`
// =========+ Create a LexicalLookupAPI object +==
LexicalLookupAPI look = new LexicalLookupAPI(args);

// ===================+ Chunk the file +==
Document aDocument = look.processDocument(aFile);

List terms = aDocument.getLexicalElements();
LexicalElement aTerm = null;

// ===========+ Print the LexicalElements out +==
for (Iterator i = terms.iterator(); i.hasNext(); ) {
    aTerm = (LexicalElement) i.next();
    System.out.println(aTerm.toPipedString());
}
• This is LVG’s fruitful variants index available as an API within the textTools
• Is used to generate variants from noun phrases extracted from documents

```java
gov.nih.nlm.nlp.lexicon.VariantLookup
```

**Constructor Summary**

```java
```

**Method Summary**

```java
Variant[] find( String pTerm )
```

```java
Variant[] find( String pTerm, int pCats, int pVarTableType )
```
• Assigns Parts of Speech (POS) to words in text
• NP parsers need terms with Parts of Speech assigned to determine phrase breaks and head assignment
• Includes LexicalLookup
## Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>TaggerClientMain</td>
<td><code>gov.nih.nlm.nls.utils.GlobalBehavior pSettings</code></td>
</tr>
<tr>
<td>TaggerClientMain</td>
<td><code>String[] args</code></td>
</tr>
</tbody>
</table>
gov.nih.nlm.nlp.taggerservices

**Interface TaggerInterface**

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag( Sentence pSentence )</td>
</tr>
</tbody>
</table>

**Class gov.nih.nlm.nlp.taggerservices.TaggerFactory**

| static TaggerInterface | build(GlobalBehavior pSettings ) |

**NLPRegistry.cfg: Example Contents:**

-043|--tagger|String|medpostskr|Name of tagger hooked in
Retinoblastoma

What is retinoblastoma?
Retinoblastoma is a rare type of eye cancer that develops in the retina, which is the part of the eye that detects light and color. Although this disorder can occur at any age, it usually develops in young children.
**Abstract:** The in-vitro antioxidant activity of natural (essential oils, vitamin E) or synthetic substances (tert-butyl hydroxyanisole (BHA), Trolox) has been evaluated by monitoring volatile carbonyl compounds released in model lipid systems subjected to peroxidation. The procedure employed methodology previously developed for the determination of carbonyl compounds as their pentafluorophenylhydrazine derivatives, which were quantified with high sensitivity by means of capillary gas chromatography with electron-capture detection. Linoleic acid and sunflower oil were used as model lipid systems. Lipid peroxidation was induced in linoleic acid by the Fenton reaction (Fenton L 1:37 °C; 12 h) and in sunflower oil by heating in the presence of O₂ (220 °C; 2 h).
MetaMap Transfer (MMTx) Display Mappings

// ======================+ Display Phrase and Concepts +==
String displayPhrase( Phrase aPhrase ) throws Exception {

// =================================+ Get the Mappings +==
List finalMappings = aPhrase.getFinalMappings();

if ( finalMappings != null ) {
    Iterator mappingIterator = finalMappings.iterator();

    // ===================+ Iterate through the Mappings +==
    while (mappingIterator.hasNext()) {
        FinalMapping aMapping = (FinalMapping) mappingIterator.next();
        System.out.println( aMapping );
    }
}
}
// ==========================+ Create a MMTxAPI object ====
MMTxAPI mmtx = new MMTxAPI();

// =============================+ Analyze the Sentence ====
Sentence aSentence = mmtx.processSentence("Insomnia is a symptom of a sleep disorder");

Iterator phraseIterator = aSentence.getPhrases().iterator();

// ======================+ Iterate through the Phrases ====
while (phraseIterator.hasNext()) {
   Phrase aPhrase = (Phrase) phraseIterator.next();

   System.out.println(displayPhrase(aPhrase));
}

// displayPhrase method
Phrase: "non-hodgkin's lymphoma"

Meta Candidates (5)

1. **1000 Lymphoma, Non Hodgkin's** [Neoplastic Process]
2. **861 hodgkin's lymphoma (HODGKINS DISEASE)** [Neoplastic Process]
3. **812 Lymphoma (Germinoblastoma)** [Neoplastic Process]
4. **812 Lymphoma** [Neoplastic Process]
5. **805 NON (NON Mouse)** [Mammal]

Meta Mapping (1000)

1. **1000 Lymphoma, Non Hodgkin's** [Neoplastic Process]
The SPECIALIST dTagger (A POS Tagger)

specialist.nlm.nih.gov

- Introduction
- Features
- Lexical Lookup
- Handling Unknowns
- Training
- Tagging
- Updating
- Error Analysis
- Work Still to Do
Motivation

Why another POS tagger?

• SPECIALIST Lexicon
• Arbitrary tag set
• Supervised and unsupervised training and updating
• True multi-word (term based) tagger
• Generalizable to other languages
Features

• Tag set specified as a configurable file
  Just make sure Lexicon/tagset/corpus use the same tags.

• Lexical Information (from .lex files)
  SPECIAST lexicon
  Pseudo lexicon (number words, roman numerals)
  Verbs as adjectives
  Local lexicon
Features (2)

- Lexical lookup, pattern recognition to tokenize into terms
- Unknown words handled with several strategies
- Hidden Markov Model, Viterbi Model used for training and tagging
- Probability of correct tag assignment reported back
Lexical Lookup

Longest spanning matches from Lexicon

.. on diabetes gravidarum diagnosis and treatment

on diabetes mellitus

diabetes Atherosclerosis Intervention Study

diabetes gravidarum
diabetes insipidus

on diabetes gravidarum diagnosis and treatment
Handling Unknowns

• Overall probability of the next word being an unknown word gathered for (open class) words with low frequency.

• Suffix statistics gathered from annotated corpus or from Lexicon (for unsupervised training) for open class terms.
Handling Unknowns: Shape Identification

Fifty two trials were identified which fulfilled wordnum noun | aux | verb | pron | verb |

In 17 trials with placebo groups prep | num | noun | prep | noun | noun |

... cure rate of placebo preparations was 30% |... verb | noun | prep | noun | noun | aux | percent |

... of 10 weeks | ( | range | 4 | to | 24 weeks | ) |

... prep | unitOfMeasure | lp | noun | num | prep | unitOfMeasure | rp |

... myocardial | scintigraphy | (P=0.02) |

... adj | noun | levelOfSignificance
Handling Unknowns: Shape Identification (2)

Units of Measure
Level of Significance
Experiment Size
Number
Real Number
Word Number
Telephone Number

Range
EmailAddress
Date
URL
Fraction
Percent
Glob

Equation
Address
Chemical
Gene Name
Proper Name
Name
Delimiter

*Next or future release
Unsupervised Updating and Training

- Unsupervised Updating
  With a prior model and lexicon and an un-annotated corpus
  Even 10 hand annotated sentences as the initial model gives a big boost

- Unsupervised Training
  With only the lexicon and no prior model
  Suffix statistics gleaned from lexicon
dTagger
Available Programs

• TrainWithTaggedText
• Tag
• UpdateWithUntaggedText
• TrainWithUntaggedText
• MorphologyDiscovery
• AnalizeTaggedCorpus (next release)
Unsupervised Updating and Training

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>--dirName=</td>
<td>Where the input files will be found</td>
<td>the current directory</td>
</tr>
<tr>
<td>--fileName=</td>
<td>The input file or input file pattern</td>
<td>standard input</td>
</tr>
<tr>
<td>--modelName=</td>
<td>Saved Hidden Markov model name</td>
<td>“default”</td>
</tr>
<tr>
<td>--Rev=</td>
<td>Revision number useful for tracking multiple updates to a model</td>
<td>“0”</td>
</tr>
</tbody>
</table>
## Unsupervised Updating and Training

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>--corpusName=</td>
<td>This is echoed out in output stats</td>
<td>“MedPost”</td>
</tr>
<tr>
<td>--sentenceIDMarker</td>
<td>Sentence ID in the training corpus.</td>
<td>“P”</td>
</tr>
<tr>
<td>--overwrite</td>
<td>Overwrite the current rev’s probs when updating</td>
<td>false</td>
</tr>
</tbody>
</table>
Error Analysis

• Verbs tagged as nouns
• Adj/noun and noun/adj’s
• Odd usages
• Human tagging inconsistencies
• Conflicts with what the lexicon says
The dTagger Tag Class

**Tag Constructor Summary**

| Tag(GlobalBehavior pSettings ) |

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>processCollection(Collection pCollection )</td>
</tr>
<tr>
<td>void</td>
<td>processDocument(Document pDocument )</td>
</tr>
<tr>
<td>void</td>
<td>processSentence(Sentence pSentence )</td>
</tr>
<tr>
<td>double</td>
<td>tag(LexicalElement[] pTerms )</td>
</tr>
<tr>
<td>double</td>
<td>tag(Sentence pSentence )</td>
</tr>
<tr>
<td>LexicalElement[]</td>
<td>tag(String pSentence )</td>
</tr>
</tbody>
</table>
Work In Progress

• Put the latest and greatest on the website (includes textTools w/ dTagger now)

• More evaluation
  – Single vs. multi-word comparison
  – Using different tag sets

• Applied to Spanish to help build Spanish lexical resources

• “fix it, Make it better, make it faster” tasks
# Lexical Systems Group

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Contacts</strong></td>
<td></td>
</tr>
<tr>
<td>Allen Browne</td>
<td><a href="mailto:browne@nlm.nih.gov">browne@nlm.nih.gov</a></td>
</tr>
<tr>
<td>Guy Divita</td>
<td><a href="mailto:divita@nlm.nih.gov">divita@nlm.nih.gov</a></td>
</tr>
<tr>
<td>Chris Lu</td>
<td><a href="mailto:lu@nlm.nih.gov">lu@nlm.nih.gov</a></td>
</tr>
<tr>
<td>Susanne M. Humphrey</td>
<td><a href="mailto:humphrey@nlm.nih.gov">humphrey@nlm.nih.gov</a></td>
</tr>
<tr>
<td>Lexical Systems Group</td>
<td><a href="mailto:umlslex@nlm.nih.gov">umlslex@nlm.nih.gov</a></td>
</tr>
</tbody>
</table>

---

**THE LISTER HILL NATIONAL CENTER FOR BIOMEDICAL COMMUNICATIONS**

*A Research Division of the U.S. National Library of Medicine*
This Space to be filled in in Brisbane
Hidden Markov Model for POS Tagging

- The
- a
- acute
- quick
- run
- heal
- process
- run
- process
- heart

Diagram:
- States
  - Transition Probabilities
  - Emission Probabilities
- Observables
Training (1)

Hidden Markov Model

Emission counts gathered from annotated corpus

<table>
<thead>
<tr>
<th></th>
<th>Noun</th>
<th>Adj</th>
<th>Adv</th>
<th>Verb</th>
<th>Det</th>
<th>Conj</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7406</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>cause</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Training (2)

Transition counts gathered from annotated corpus

|      | Noun  | Adj  | Adv  | Verb  | Det  | Conj | ...
|------|-------|------|------|-------|------|------|-----
| Noun | 14242 | 1235 | 713  | 3231  | 3100 | 424  |     
| Adj  | 11421 | 1703 | 113  | 11    | 4    | 1    |     
| Adv  | 140   | 631  | 285  | 2014  | 359  | 9    |     
| Verb | 1273  | 758  | 794  | 97    | 19   | 3    |     
| ...  |       |      |      |       |      |      |     

Tagging

- Viterbi Algorithm:

<table>
<thead>
<tr>
<th>Do</th>
<th>viruses</th>
<th>cause</th>
<th>cancer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noun</td>
<td>97 %</td>
<td>45 %</td>
<td>97 %</td>
</tr>
<tr>
<td>Adj</td>
<td>77 %</td>
<td>52 %</td>
<td></td>
</tr>
<tr>
<td>Adv</td>
<td>23 %</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verb</td>
<td>.07 %</td>
<td>.07 %</td>
<td>.07 %</td>
</tr>
<tr>
<td>Unknown</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Do</th>
<th>viruses</th>
<th>cause</th>
<th>cancer</th>
</tr>
</thead>
<tbody>
<tr>
<td>34.92</td>
<td>verb</td>
<td>noun</td>
<td>verb</td>
</tr>
</tbody>
</table>