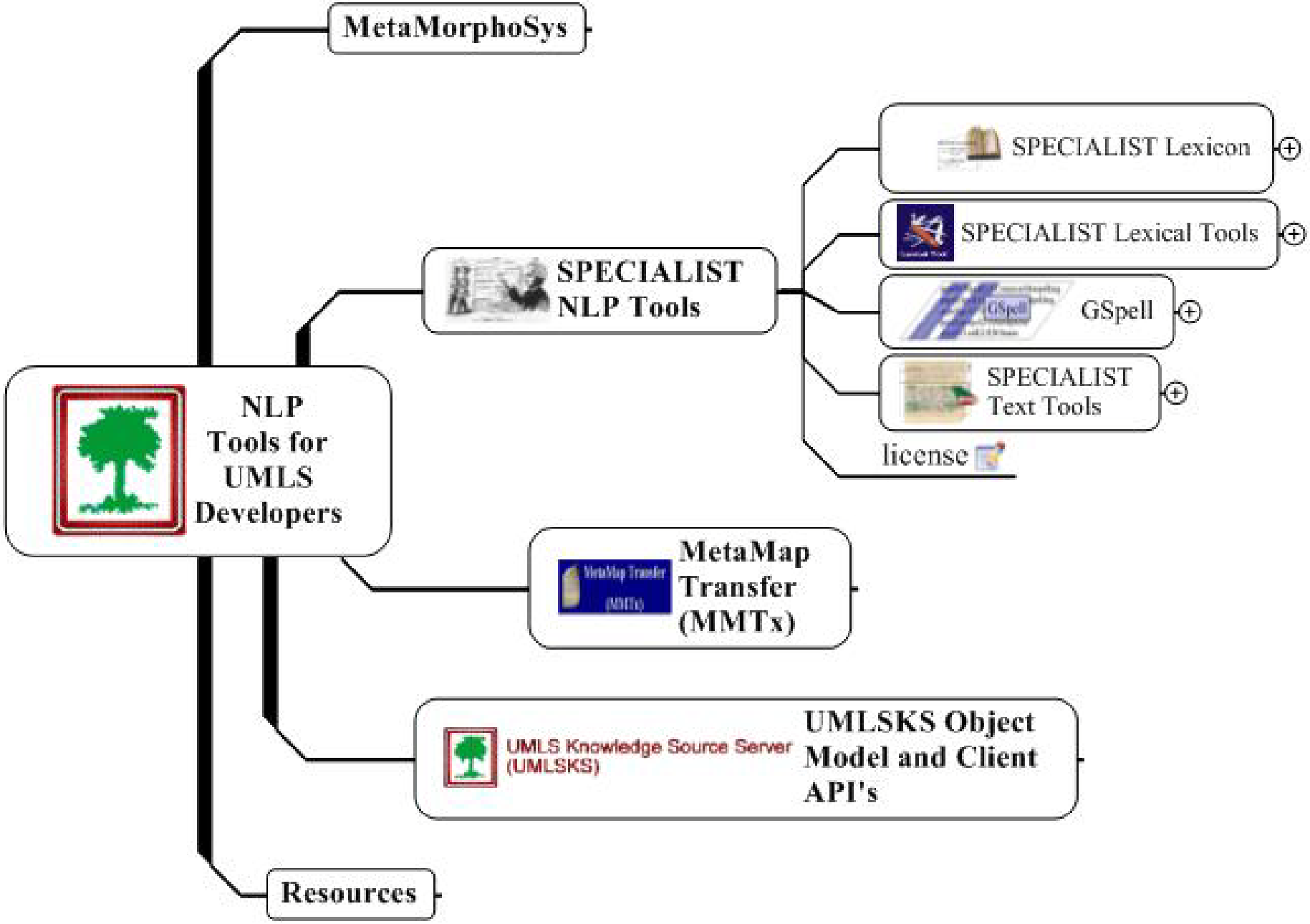
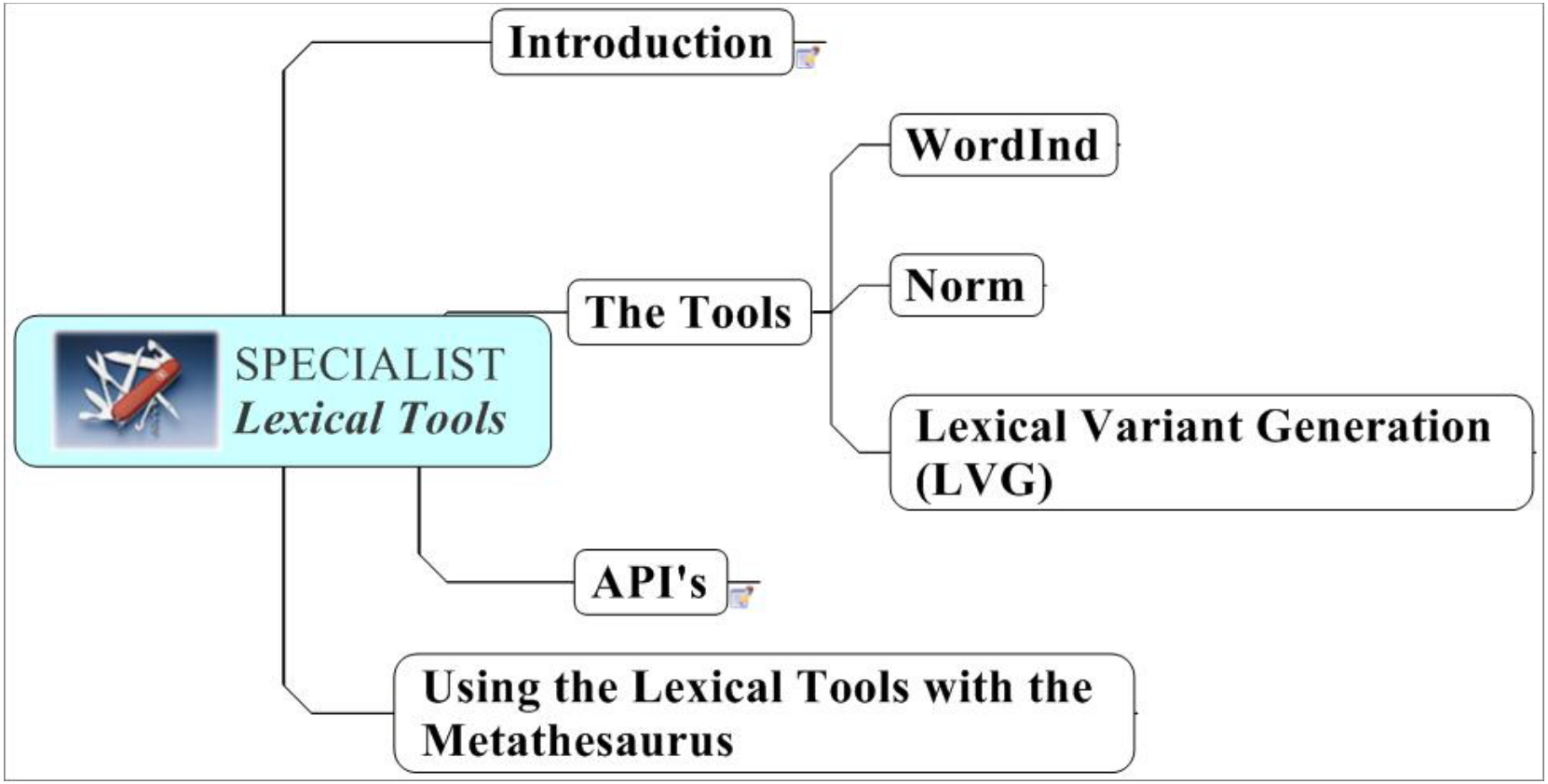

MetaMorphoSys
SPECIALIST NLP Tools
SPECIALIST Lexicon
SPECIALIST Lexical Tools
GSpell
SPECIALIST Text Tools
license
NLP Tools for UMLS Developers
MetaMap Transfer (MMTx)
UMLS Knowledge Source Server (UMLSKS)
UMLSKS Object Model and Client API's
Resources


Introduction
WordInd
Norm
The Tools
SPECIALIST
Lexical Tools
Lexical Variant Generation (LVG)
API's
Using the Lexical Tools with the Metathesaurus

Lexical Tools: Introduction

- Command line tools
  - norm
  - Lvg
  - wordInd
- Web GUI
- Pure Java Application
- Embeddable Java API’s

Lexical Tools: Introduction

- These tools are good for
  - aggressive text pattern matching
  - making word, term, phrase indexes
  - matching queries with indexed entries
  - increasing recall and/or precision

Lexical Tools: Introduction

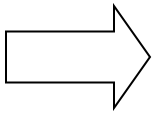
- Characteristics of all the command line tools
  - take input from the screen or a file
  - put their results to the screen or a file
  - Interpret fielded text
    - Can be told which fields contain what type of information

Lexical Tools: WordInd

Wordind is a tool to break terms into words.

It is used to take a row from a Metathesaurus table that contains a term, sentence, paragraph, story,

and break the text part of that row into it's constituent words.



wordind
is
a
tool
to
break
terms
into
words
it
is
used
to

Lexical Tools: WordInd

- Breaks words into tokens
- Passes other fields to output, untouched
- Lowercases
- Removes white space and punctuation

Lexical Tools: WordInd

Useful command line options for wordInd

-t[:Num]	Defines what field to tokenize
-f[:Num[:Num]]	Defines what fields get passed through

Lexical Tools: WordInd

> wordInd –t:7 –F:1:6

C0185495|ENG|P|L0223844|PF|**S0298948**|**Denis-Browne splint strapping**|3|

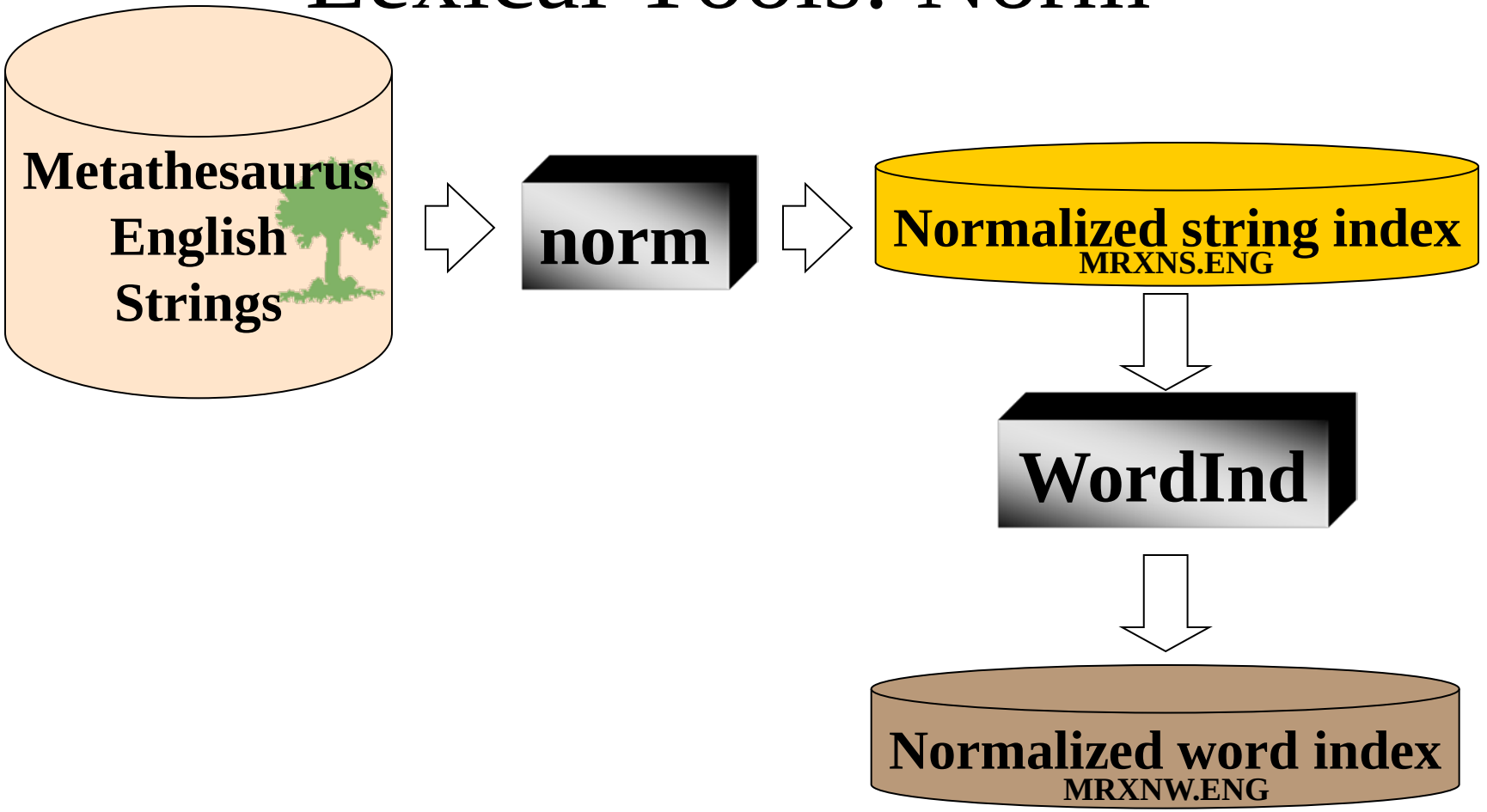
C0185495|S0298948|**denis**

C0185495|S0298948|**browne**

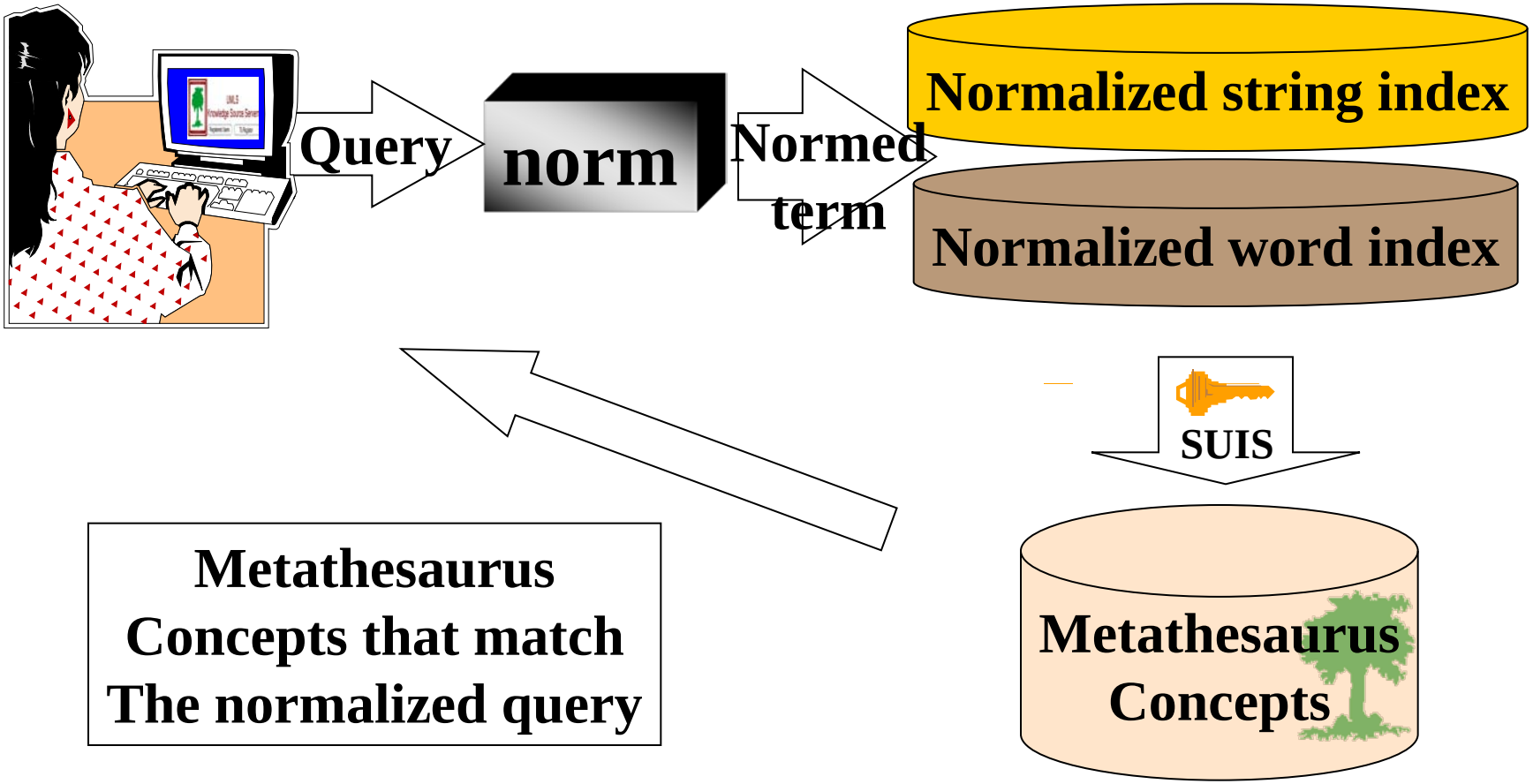
C0185495|S0298948|**splint**

C0185495|S0298948|**strapping**

Lexical Tools: Norm



Lexical Tools: Norm

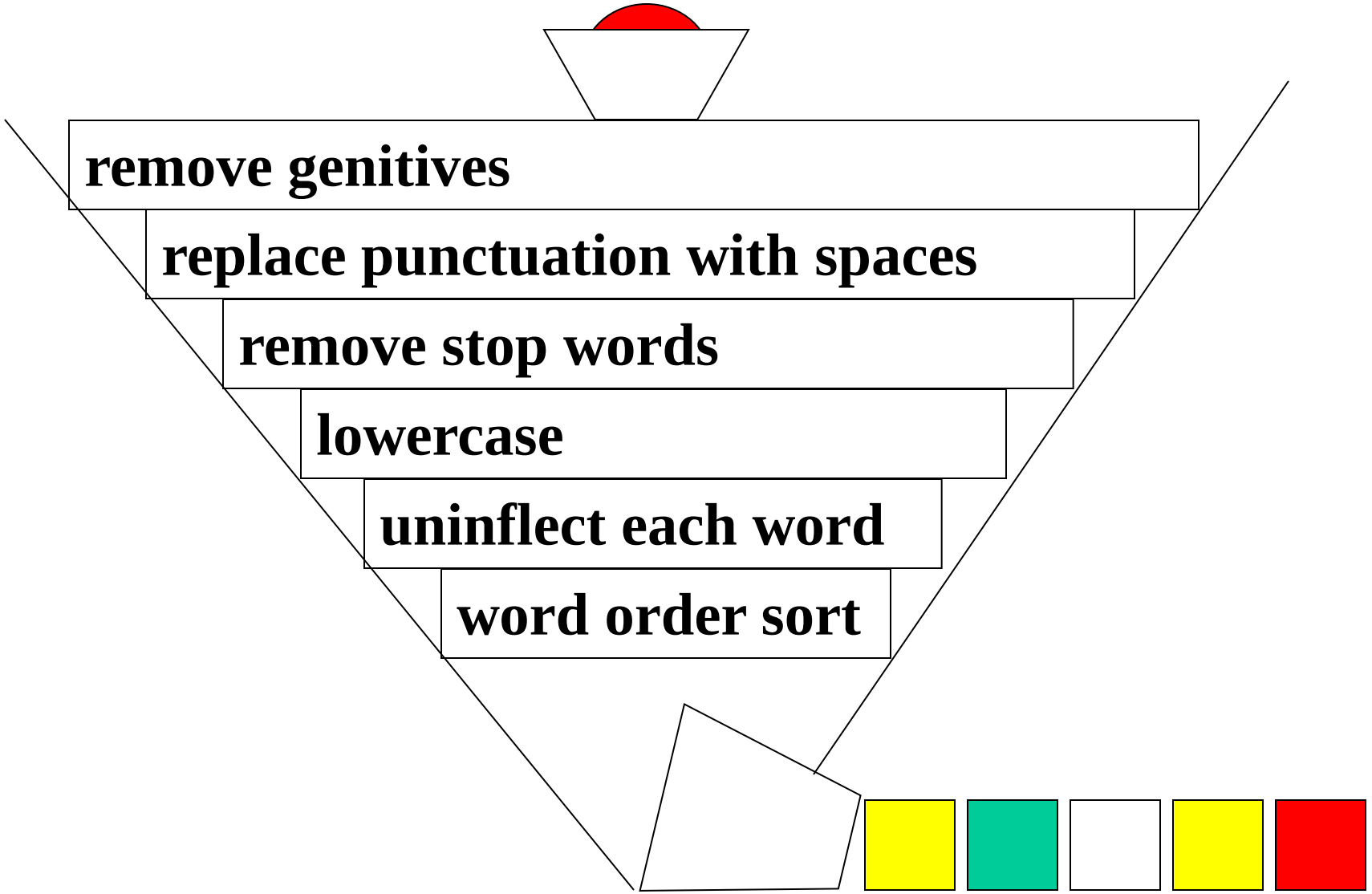


Lexical Tools: Norm

- Norm abstracts away from:
  - case
  - punctuation
  - word order
  - possessive forms
  - inflectional variation

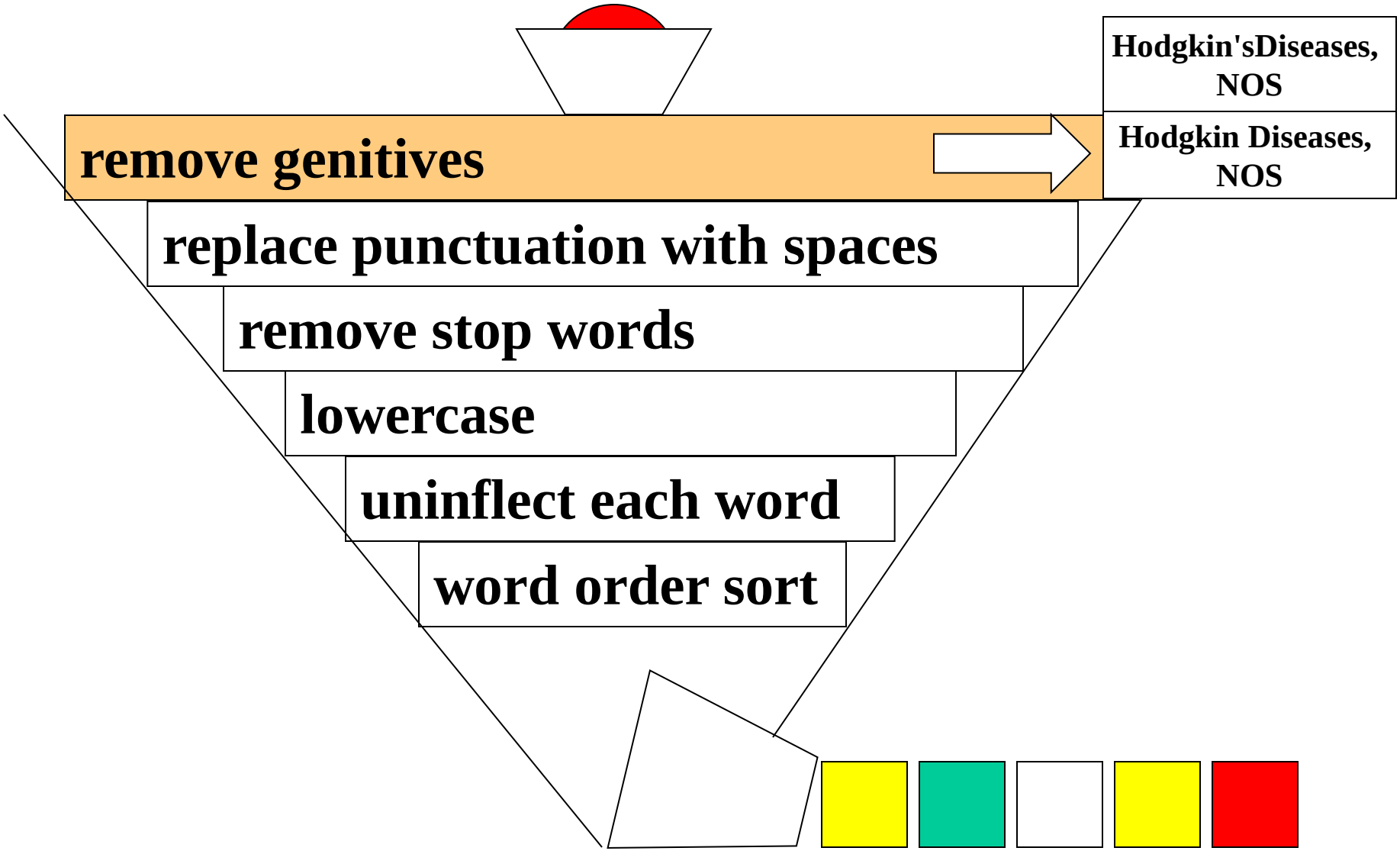
Hodgkin's Diseases, NOS

Lexical Tools: Norm



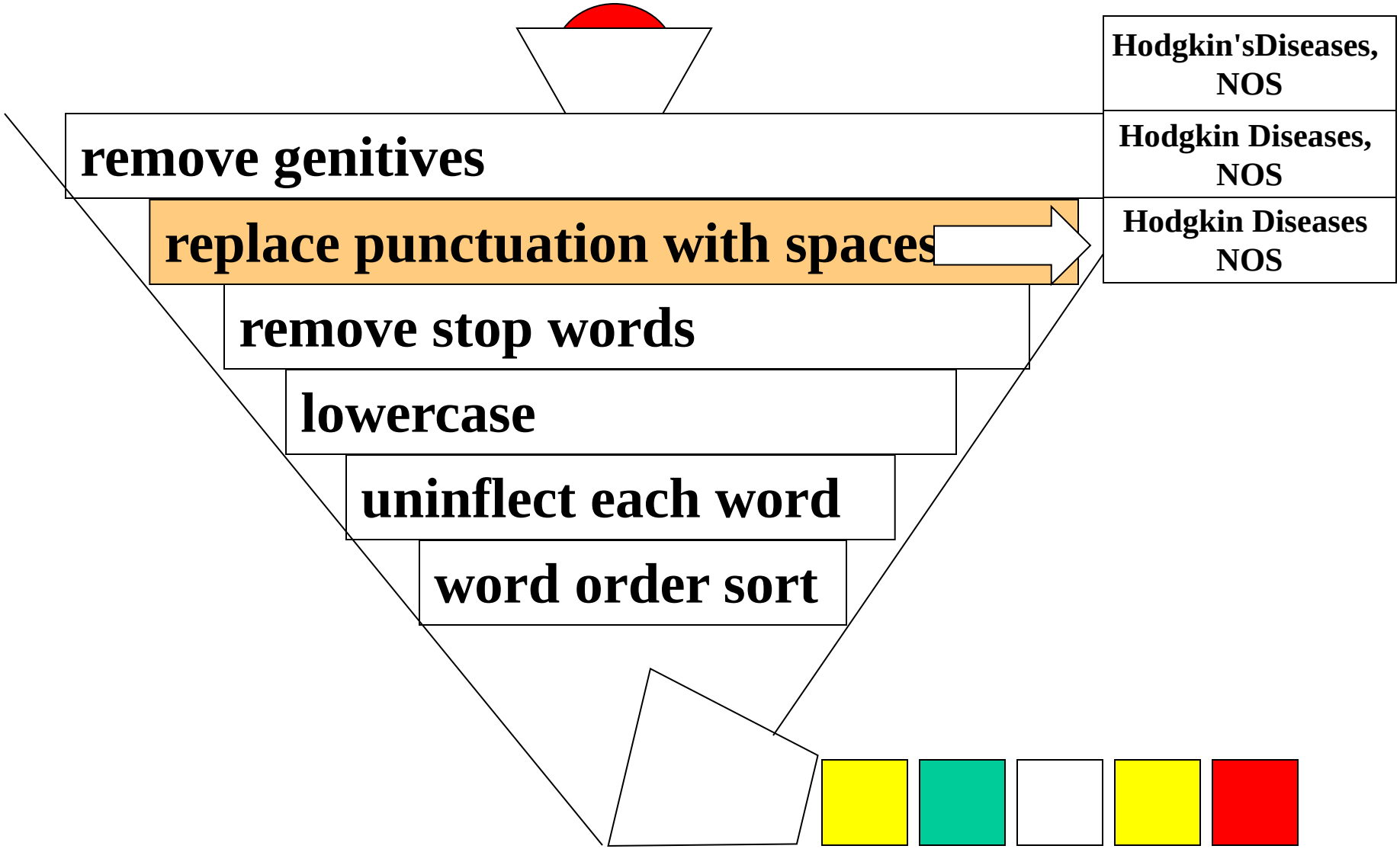
Hodgkin's Diseases, NOS

Lexical Tools: Norm



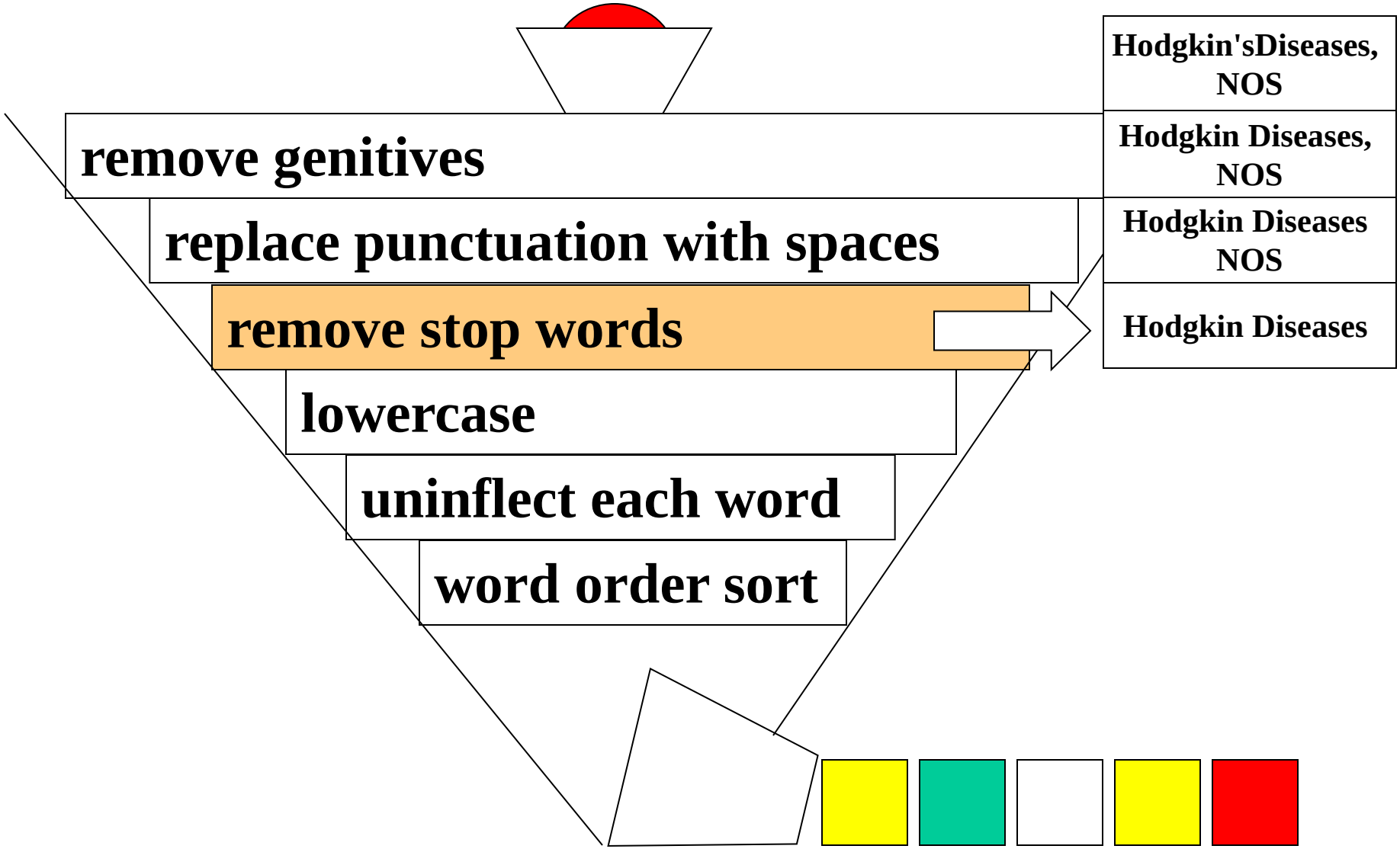
Hodgkin's Diseases, NOS

Lexical Tools: Norm



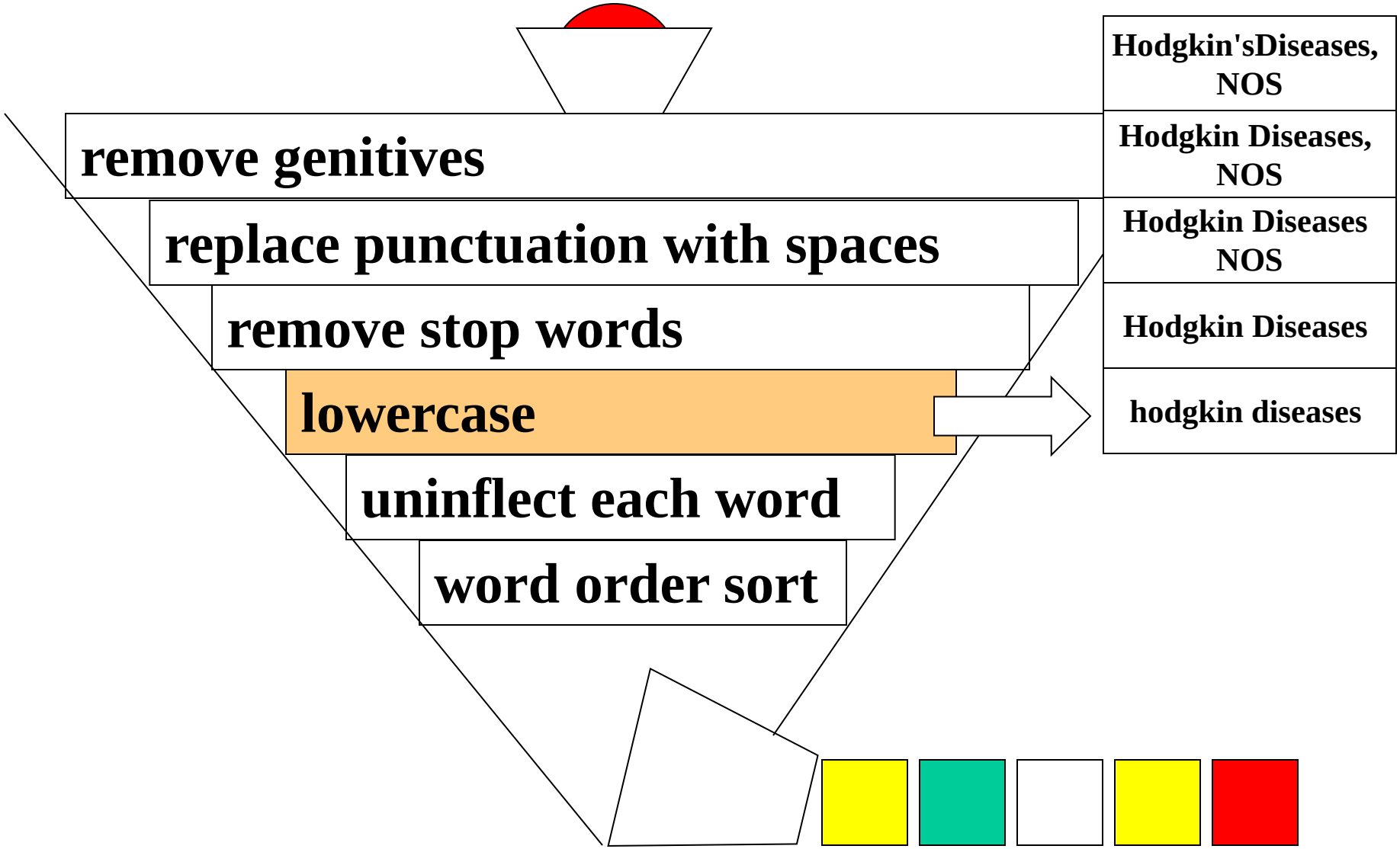
Hodgkin's Diseases, NOS

Lexical Tools: Norm



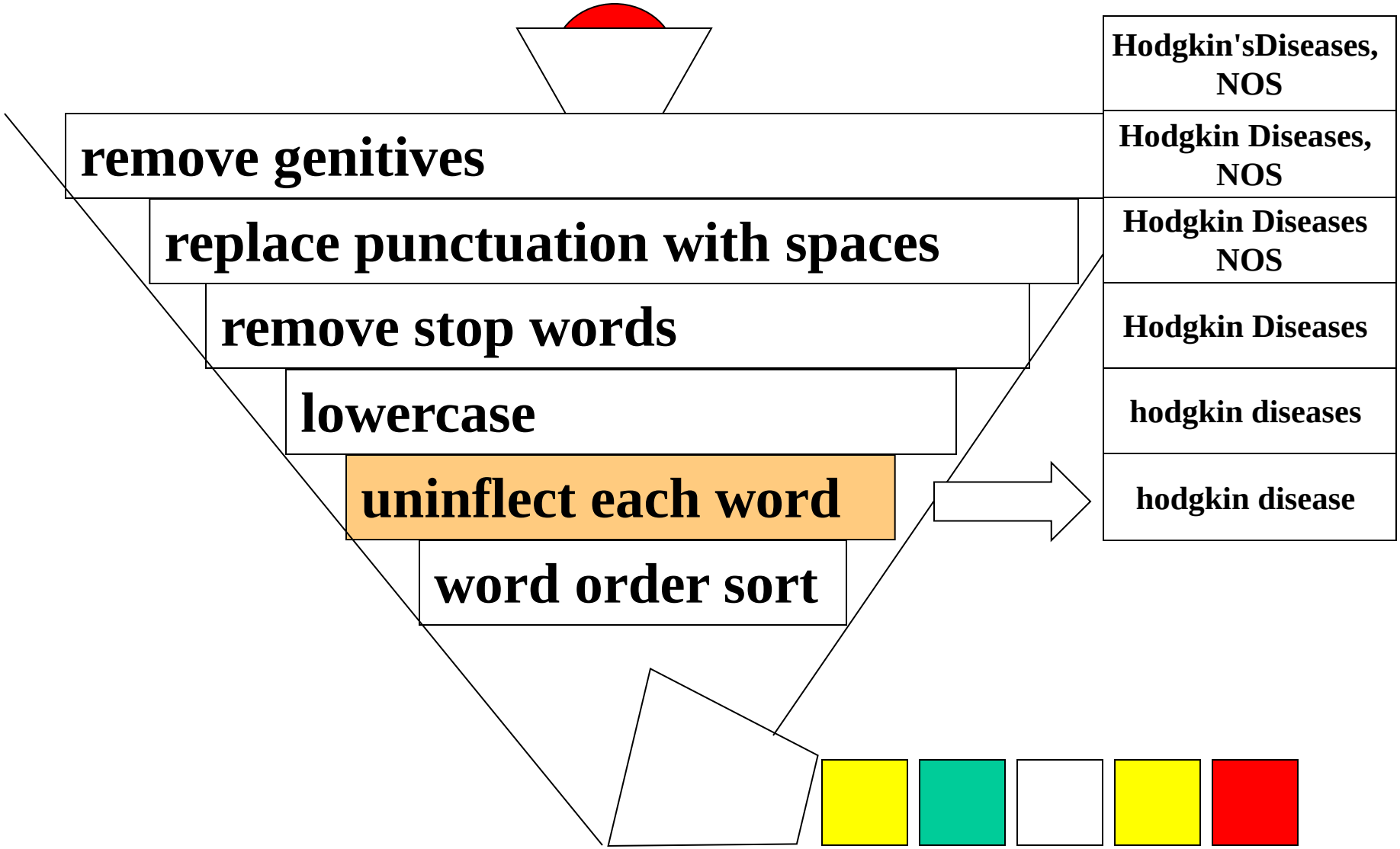
Hodgkin's Diseases, NOS

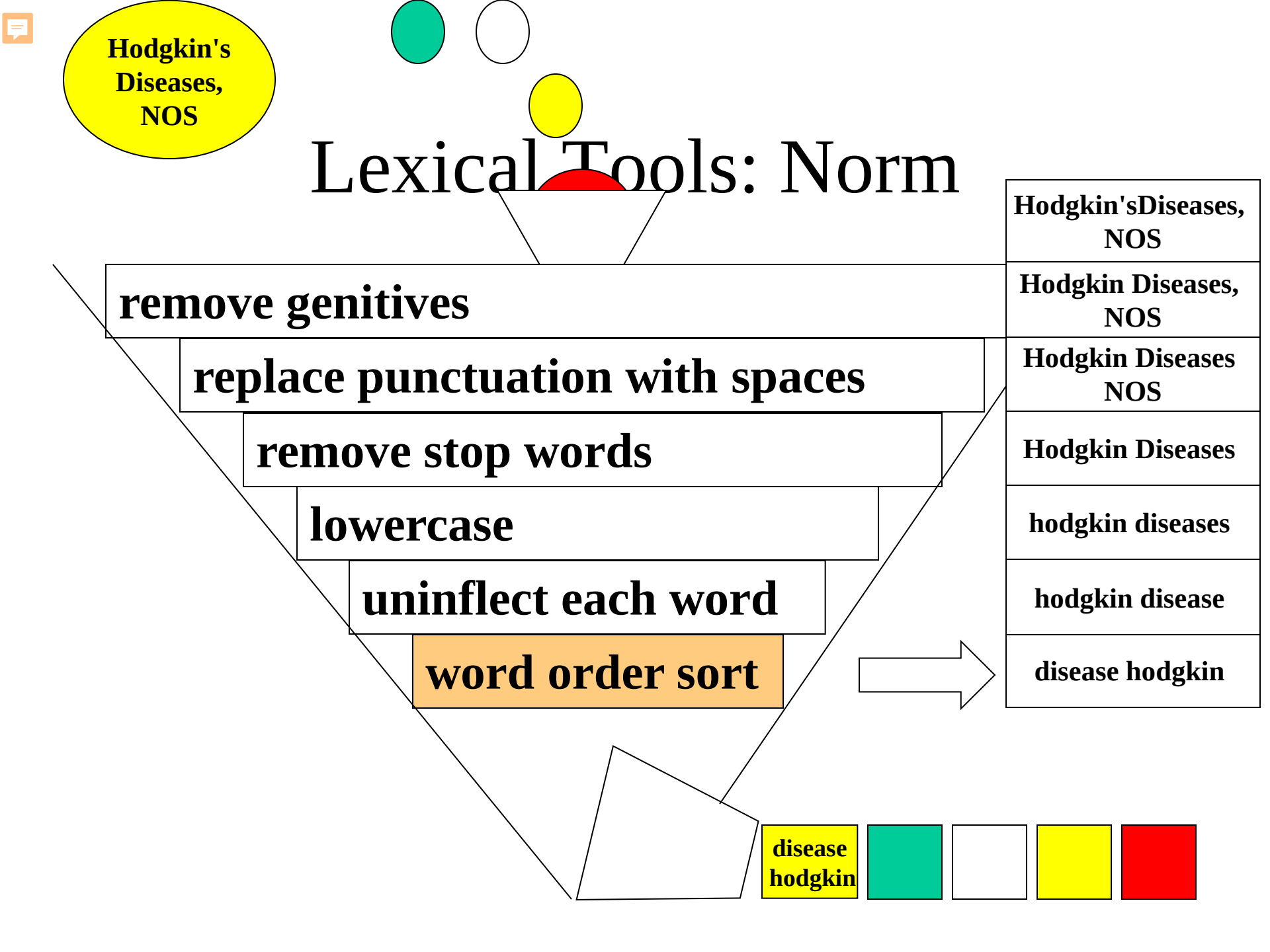
Lexical Tools: Norm



Hodgkin's Diseases, NOS

Lexical Tools: Norm



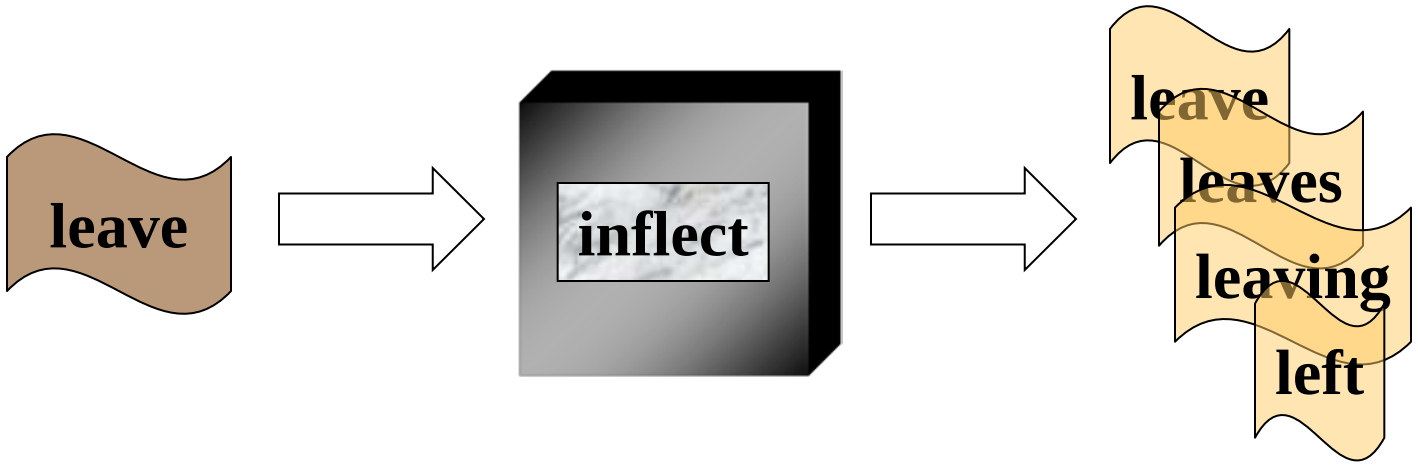

Hodgkin's Diseases, NOS
Lexical Tools: Norm
remove genitives
replace punctuation with spaces
remove stop words
lowercase
uninflect each word
word order sort
Hodgkin'sDiseases, NOS
Hodgkin Diseases, NOS
Hodgkin Diseases NOS
Hodgkin Diseases
hodgkin diseases
hodgkin disease
disease hodgkin
disease hodgkin

Lexical Tools

Lexical Tools: Flow Components

Mnemonic	Tool
A	**Return known acronyms**
a	**Return known acronym expansions**
b	**Uninflect a term**
c	**Tokenize a term into "words"**
Ct	**Retrieve the citation term**
d	**Generate derivational variants**
g	**Remove genitive**
i	**Generate inflectional variants**
L	**Retrieve category and inflection for a term**

Lexical Tools: Flows

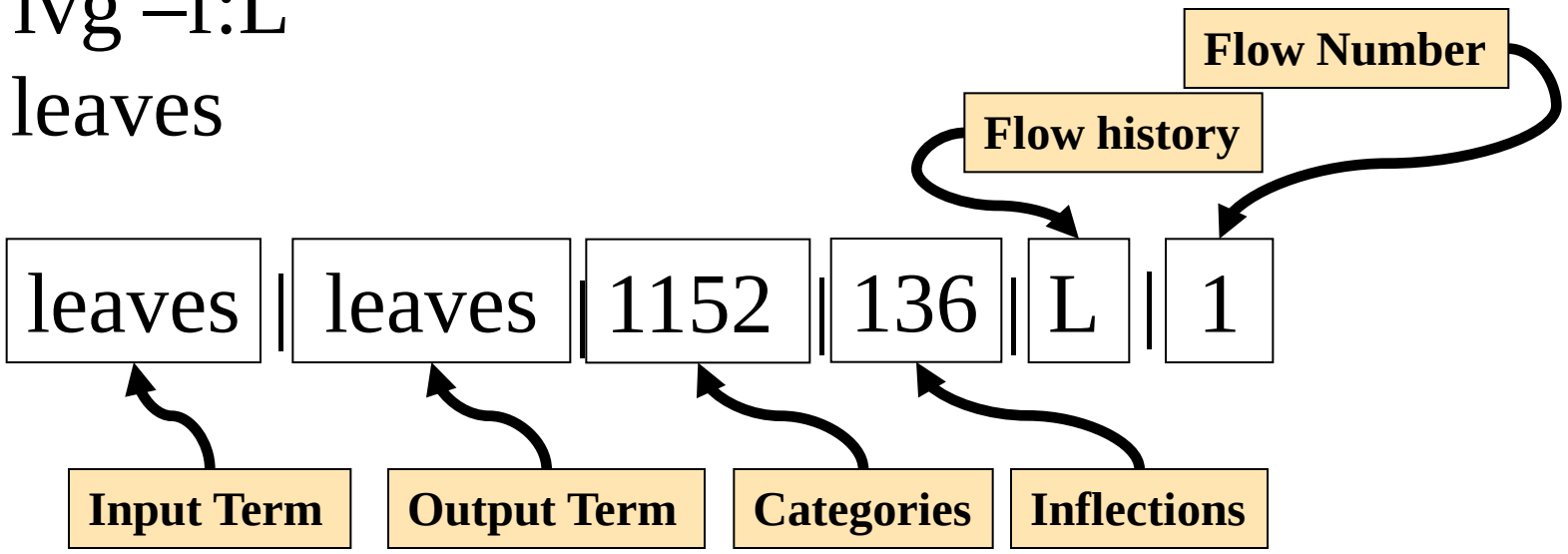


Lexical Tools: Flows

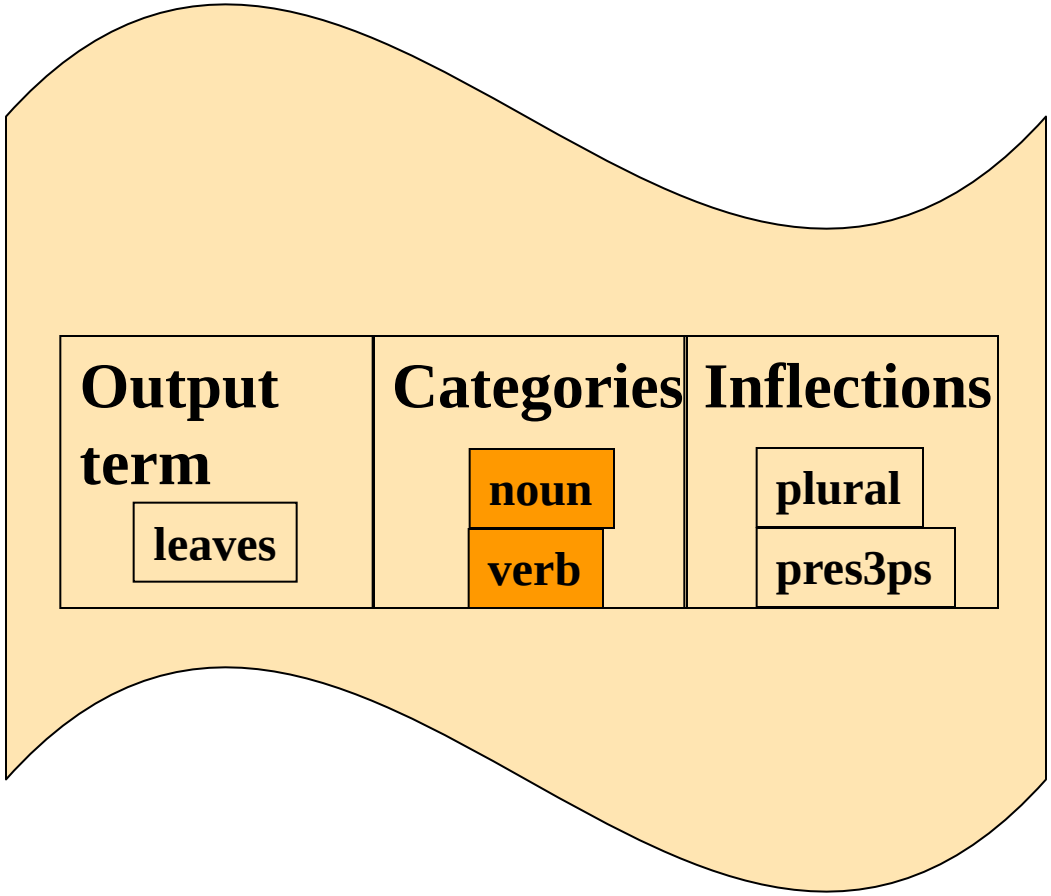
> lvg –f:i
leave
leave|**leave**|128|1|i|1|
leave|**leave**|128|512|i|1|
leave|**leaves**|128|8|i|1|
leave|**left**|1024|64|i|1|
leave|**left**|1024|32|i|1|
leave|**leave**|1024|1|i|1|
leave|**leave**|1024|262144|i|1|
leave|**leave**|1024|1024|i|1|
leave|**leaves**|1024|128|i|1|
leave|**leaving**|1024|16|i|1|

Lexical Tools: Fielded Output

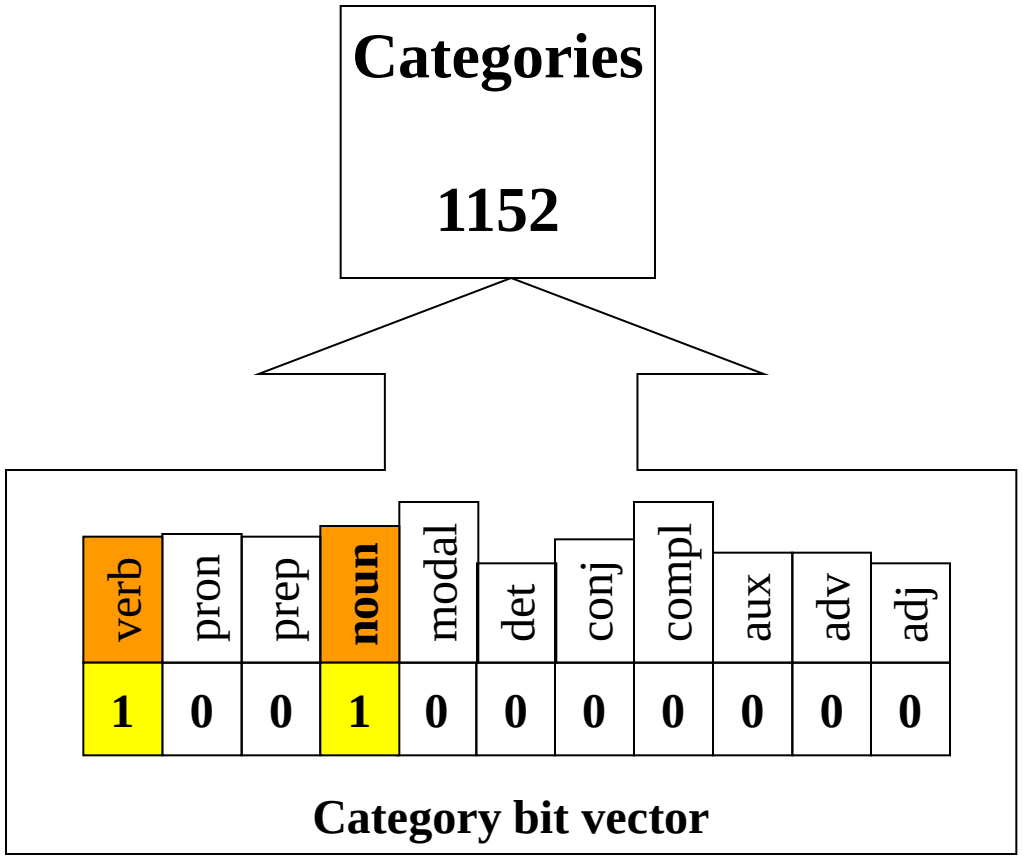
> lvg –f:L
leaves



Lexical Tools: Fielded Output



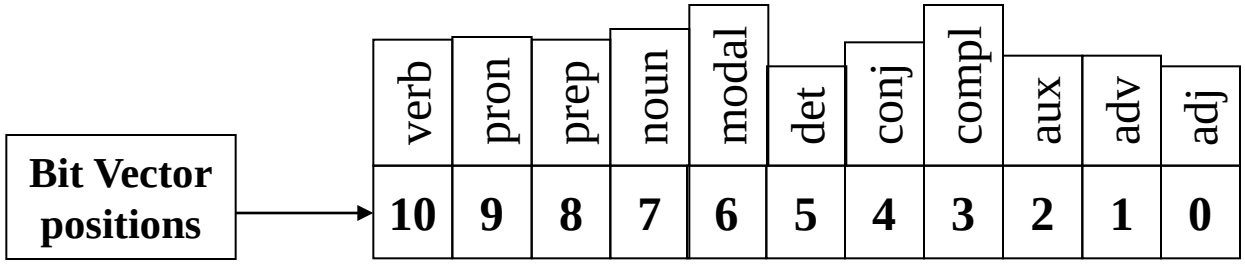
Lexical Tools: Categories



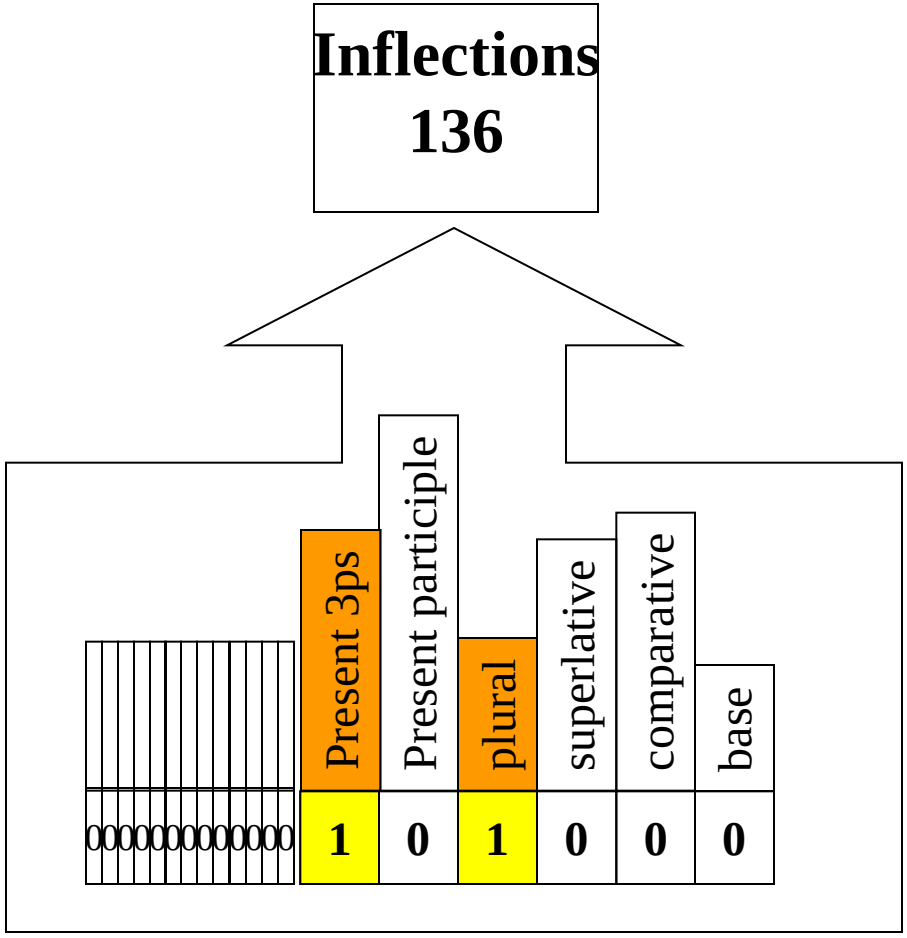
Lexical Tools: Categories

Adjective	1
Adverb	2
Auxiliary	4
Complement	8
Conjunction	16
Determiner	32

Modal	64
Noun	128
Preposition	256
Pronoun	512
Verb	1024



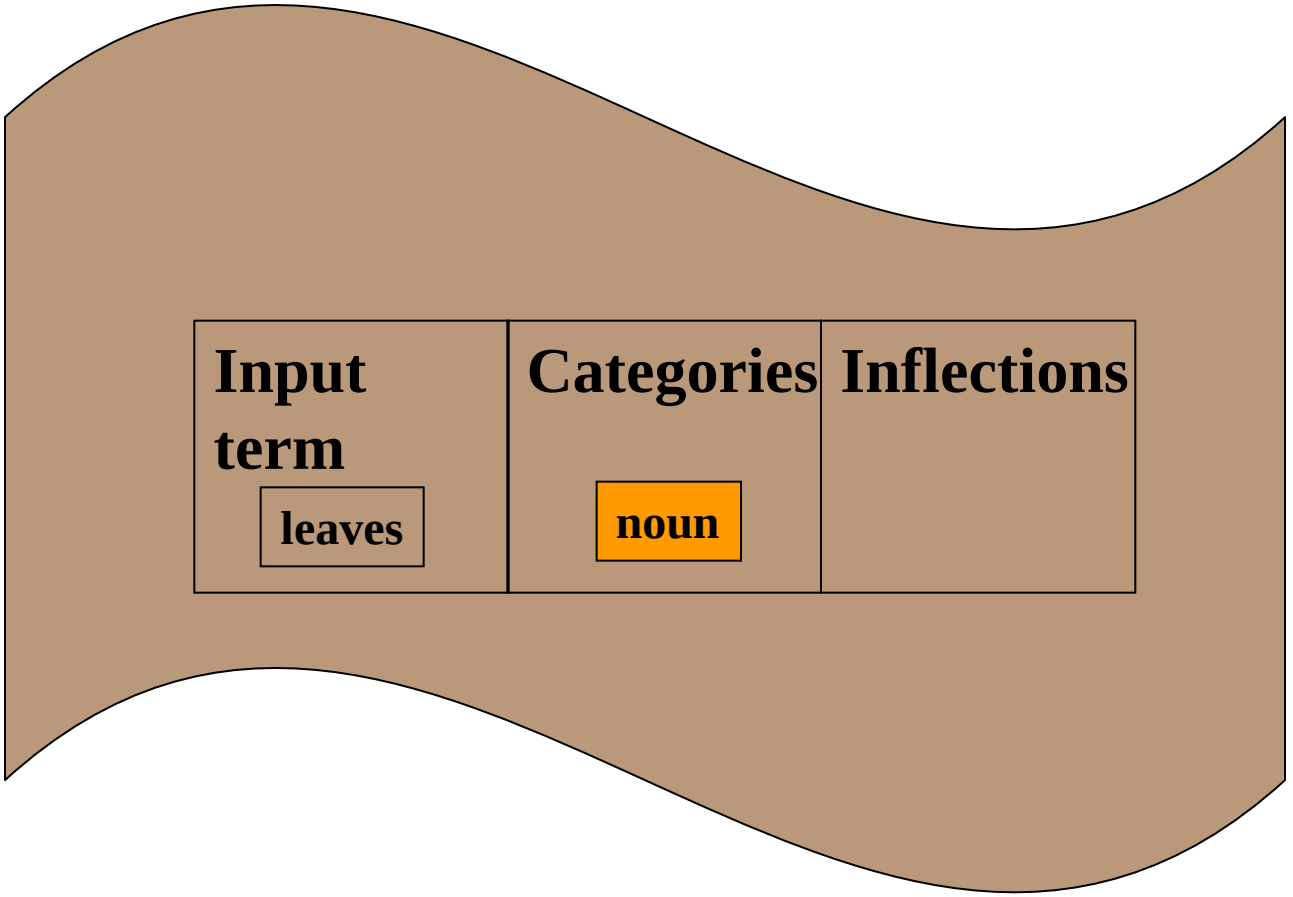
Lexical Tools: Inflections



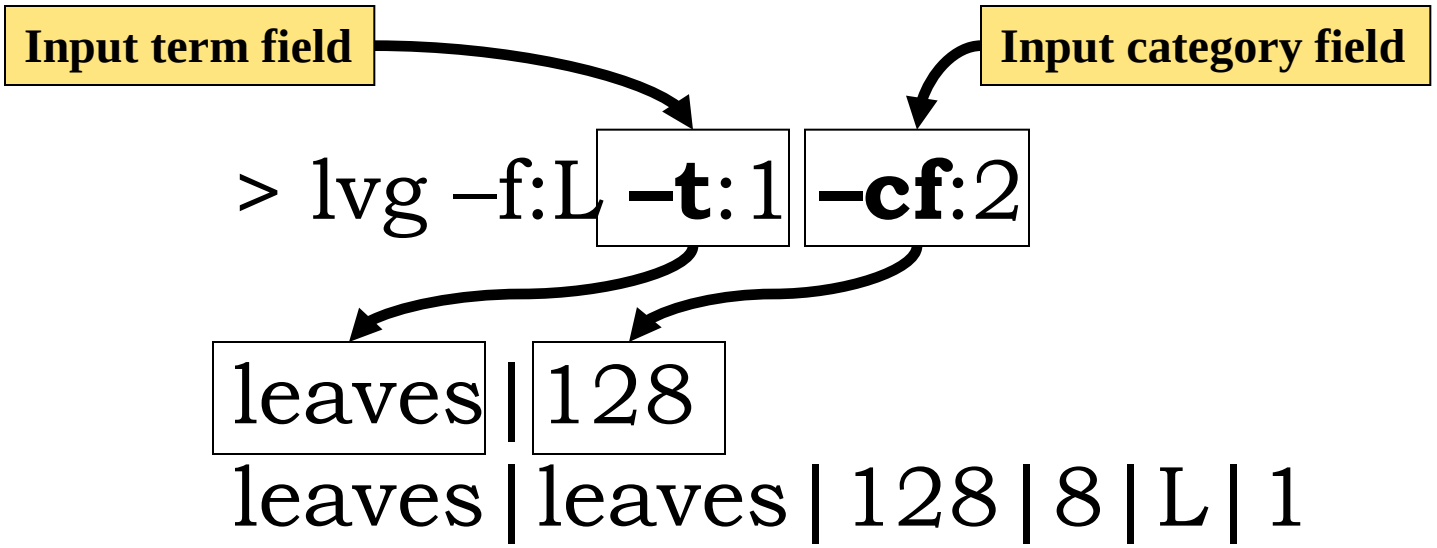
Lexical Tools: Inflections

Base	1
Comparative	2
Superlative	4
Plural	8
Present Participle	16
Past	32
Past Participle	64
Present 3rd Person Singular	128

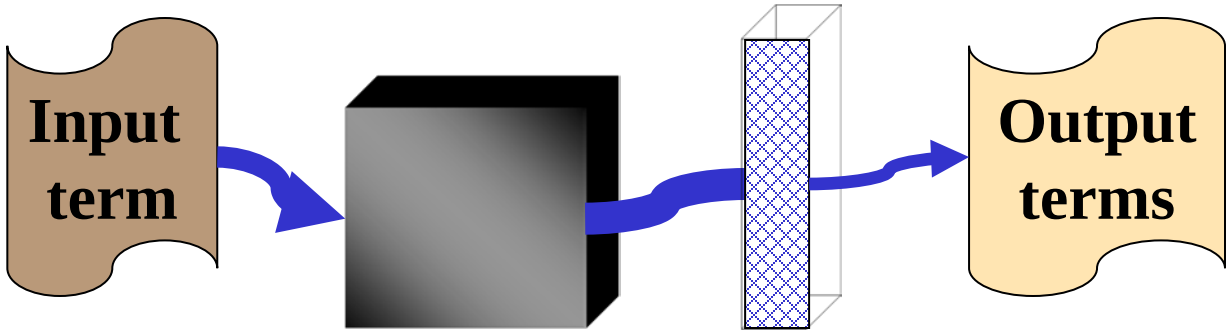
Lexical Tools: Fielded Input



Lexical Tools: Fielded Input



Lexical Tools: Post Flow Options



SC	**Show category names**
SI	**Show inflection names**
ccgi	**Mark the end of the set of variants returned**
F:*Int[:Int]*	**Specify fields for outputs**
ti	**Display the only input term in the output when using fielded input**
R:*Int*	**Restrict the number of variants returned**

Lexical Tools: Post Flow Options

Show category names

Show inflection names

> lvg -f:L **-SC -SI**

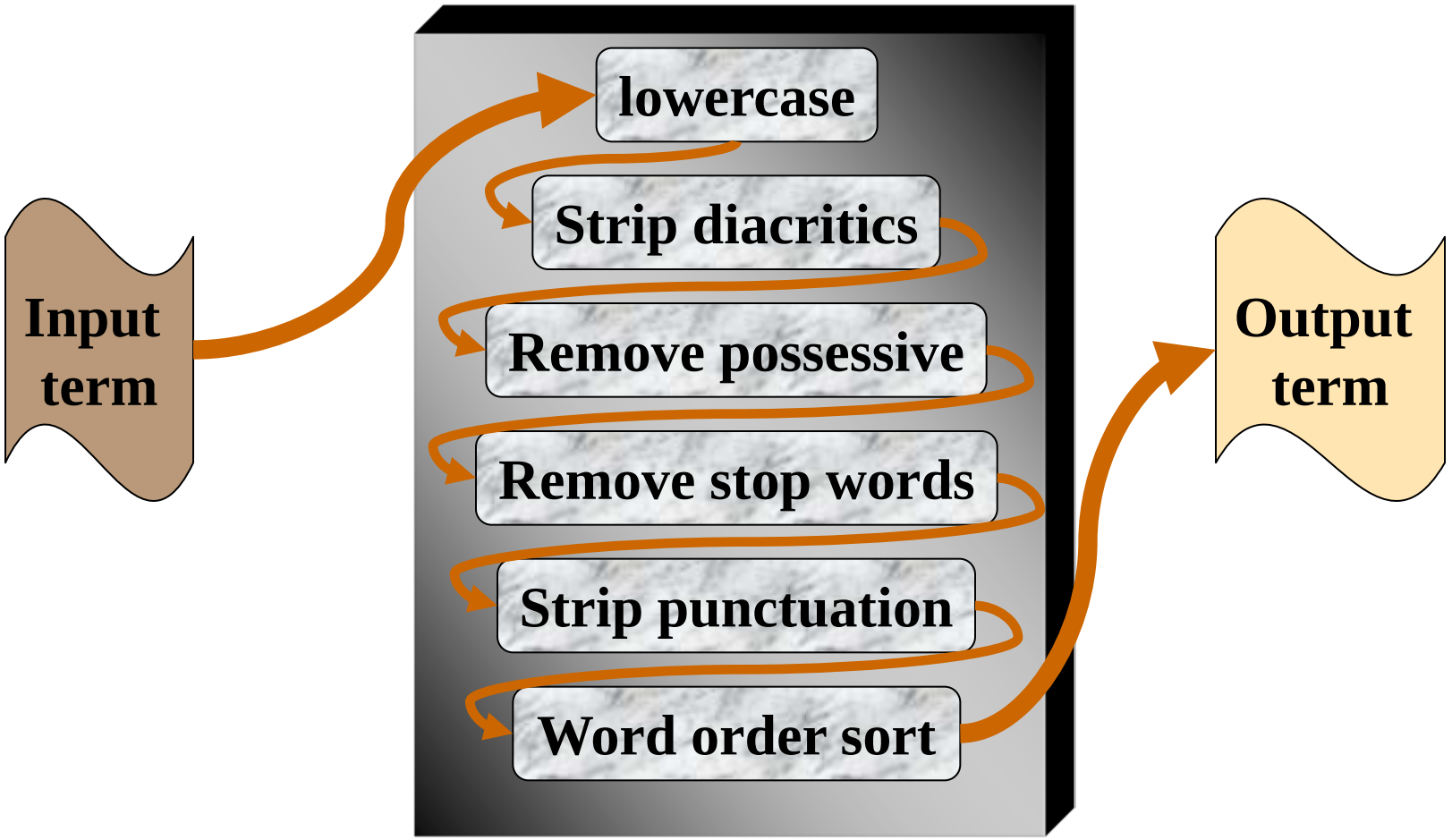
phosphoprotein

phosphoprotein|phosphoprotein|**<noun>**|**<base+singular>**|L|1|

sclerosing

sclerosing|sclerosing|**<adj+verb>**|**<base+presPart+positive>**|L|1|

Lexical Tools: A Serial Flow

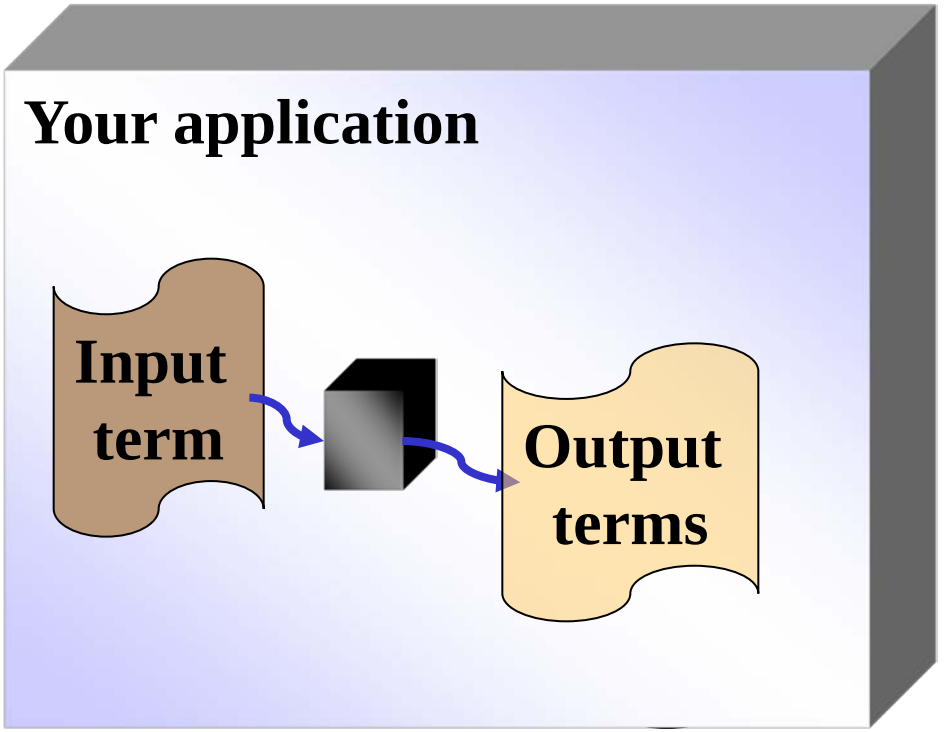


Flow components can be arranged so that the output of one is the input to another.

Lexical Tools:

Embedding These Tools into Your Application

- Classpath
- NormApi()
- LvgCmdApi()



Lexical Tools:

Embedding These Tools into Your Application

CLASSPATH = ${CLASSPATH}:
${LVG_DIR}:
${LVG_DIR}/lib/*lvg2003dist.jar:*

Lexical Tools:

Embedding Norm into Your Application

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


NormApi       normalize = new NormApi();
String          input2Norm = null;
Vector  outputFromNorm = null;
```

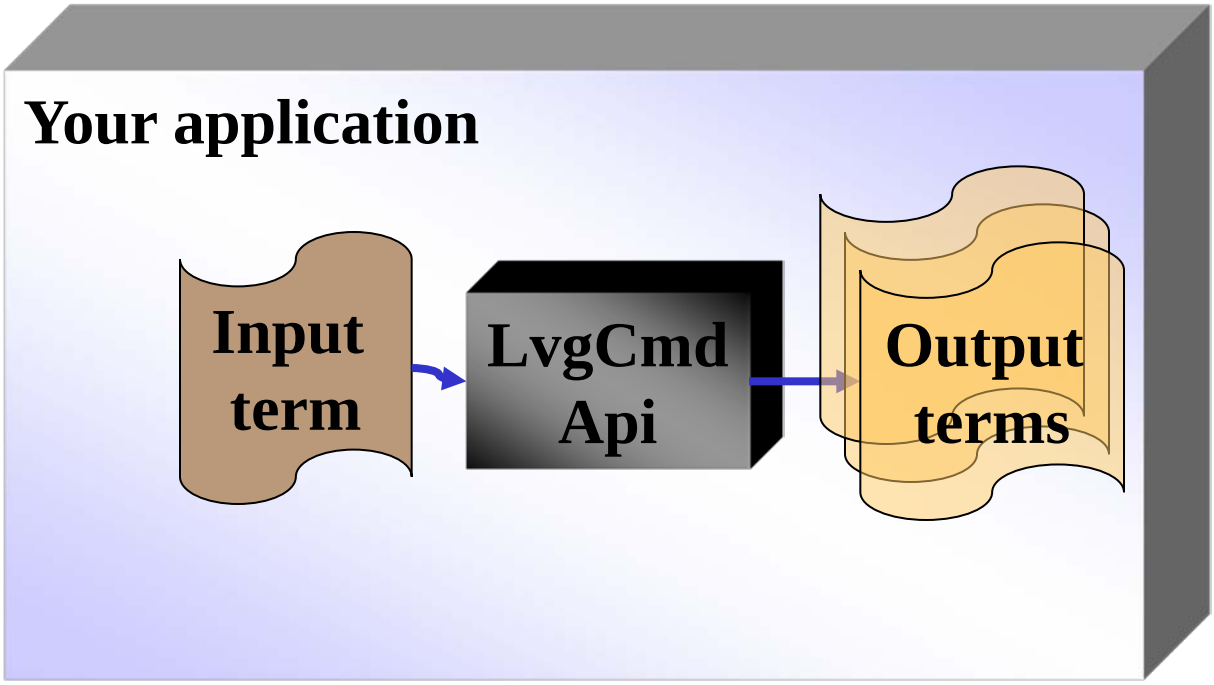
Lexical Tools:

Embedding Norm into Your Application

```
while ( (input2Norm = stdIn.readLine() ) != null ) {

   outputFromNorm= normalize.Mutate(input2Norm);

   for ( int i = 0; i < outputFromNorm.size(); i++ ) {

       System.out.println((String) outputFromNorm.get(i));
    }
}
normalize.CleanUp();
```

Lexical Tools:

Embedding Lvg into Your Application



Lexical Tools:

Embedding Lvg into Your Application

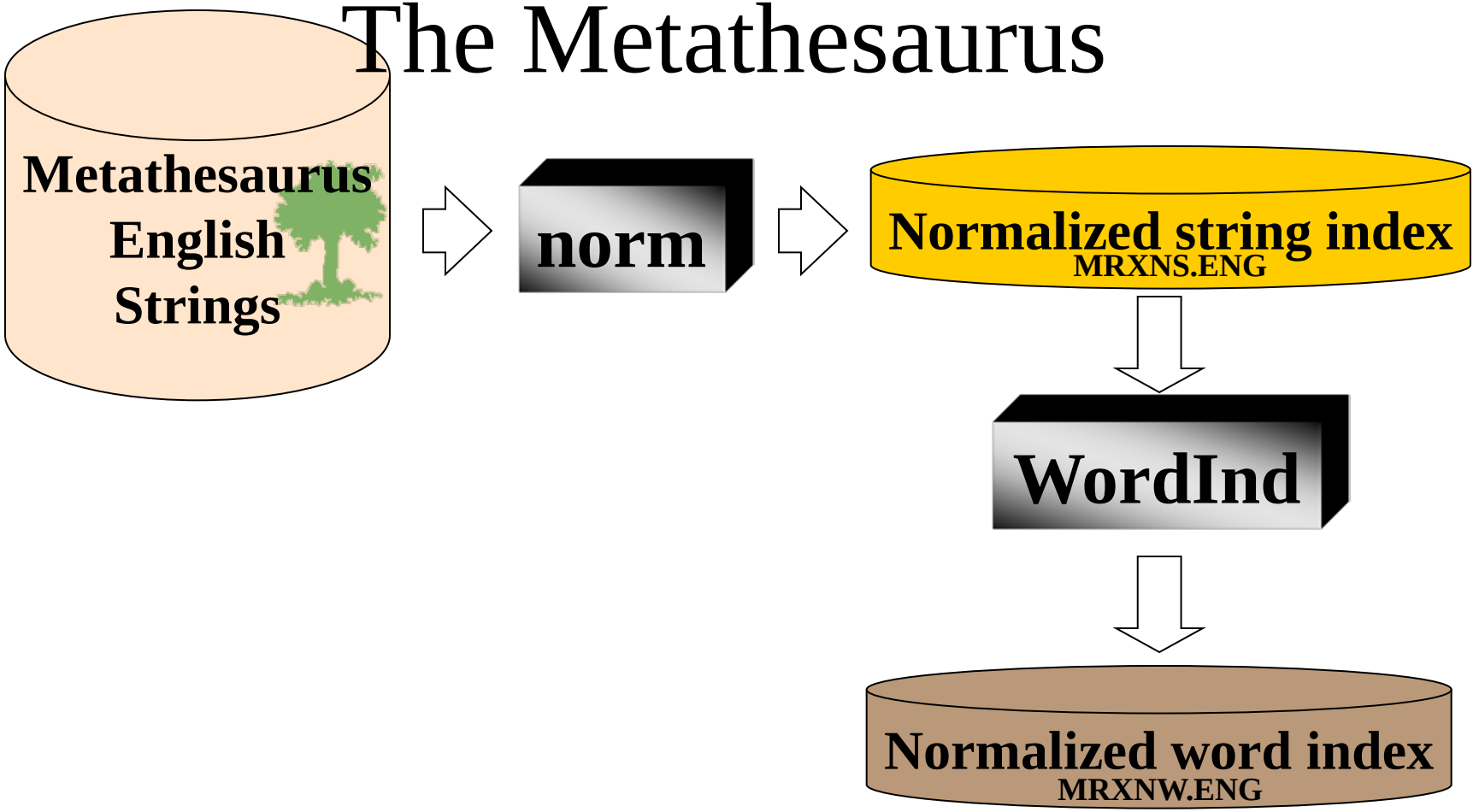
```
import gov.nih.nlm.nls.lvg.Api.*;

LvgCmdApi lvgApi = new LvgCmdApi("-f:b –CR:o –SC –SI");
String          input2Lvg = null;
Vector   outputFromLvg = null;
```

Lexical Tools:

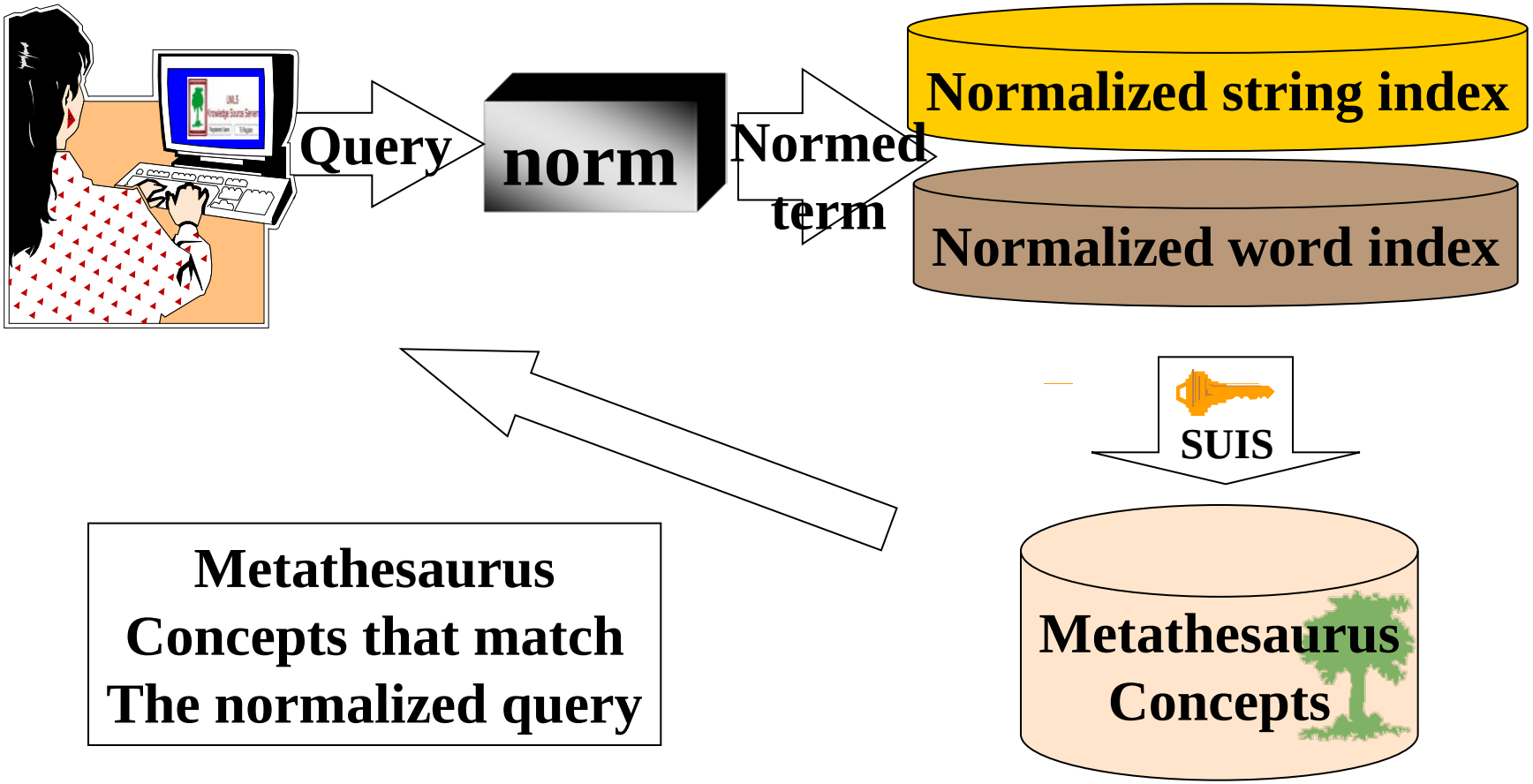
Embedding Lvg into Your Application

```
while ( (input2Lvg = stdIn.readLine() ) != null ) {
   outputFromLvg= lvgApi.MutateToString(input2Lvg);
    for ( int i = 0; i < outputFromLvg.size(); i++ ) {
       System.out.println((String) outputFromLvg.get(i));
    }
}
lvgApi.CleanUp();
```

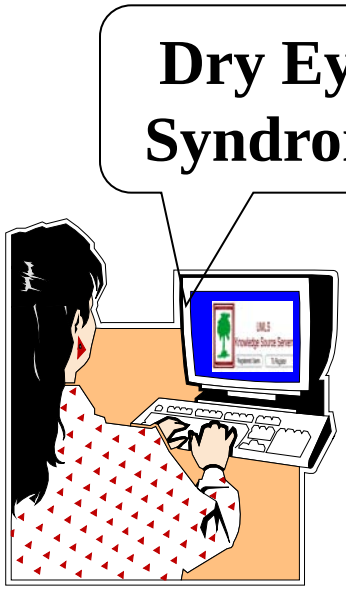
Using The Lexical Tools with The Metathesaurus



Using The Lexical Tools with The Metathesaurus

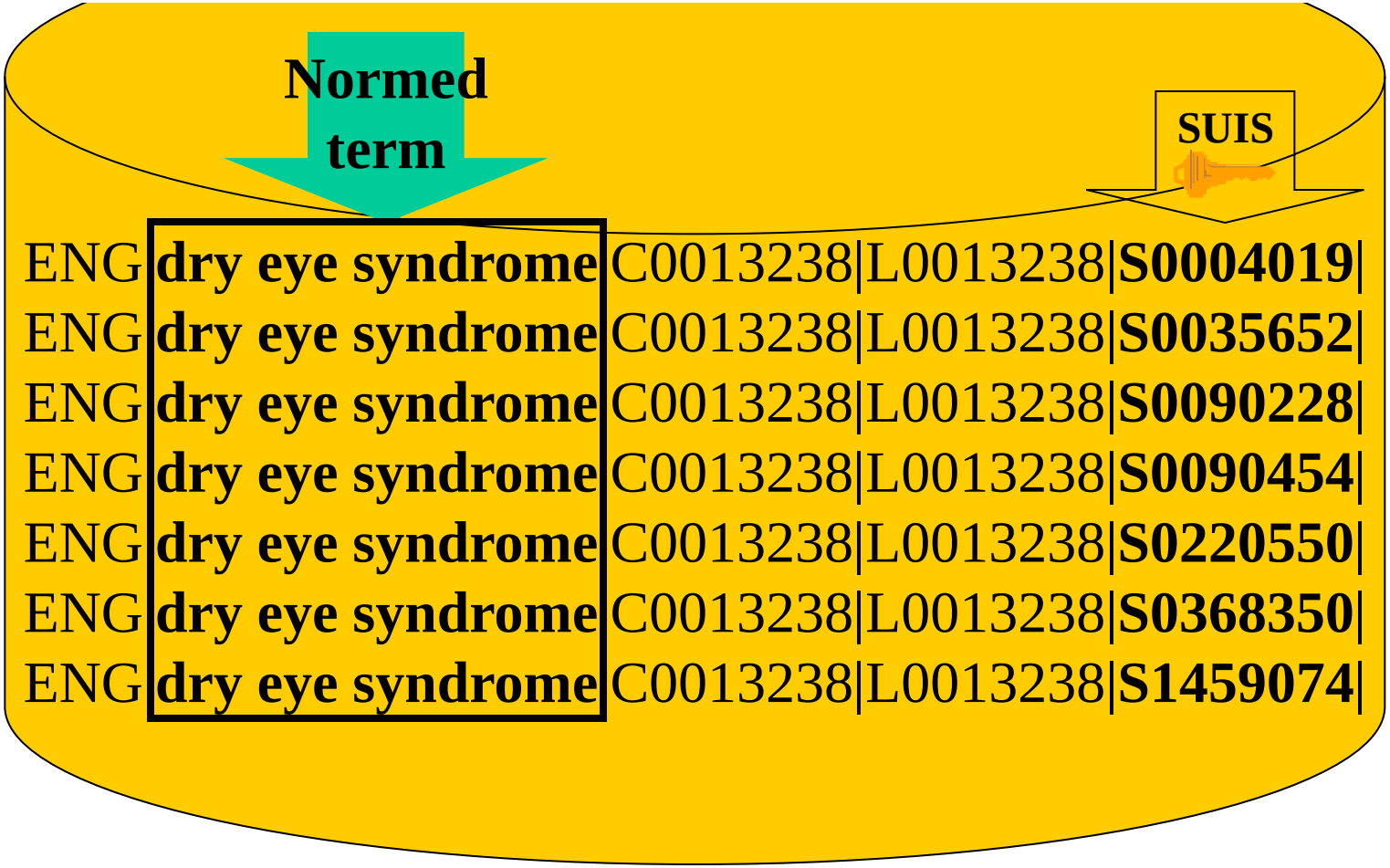


Using The Lexical Tools with The Metathesaurus

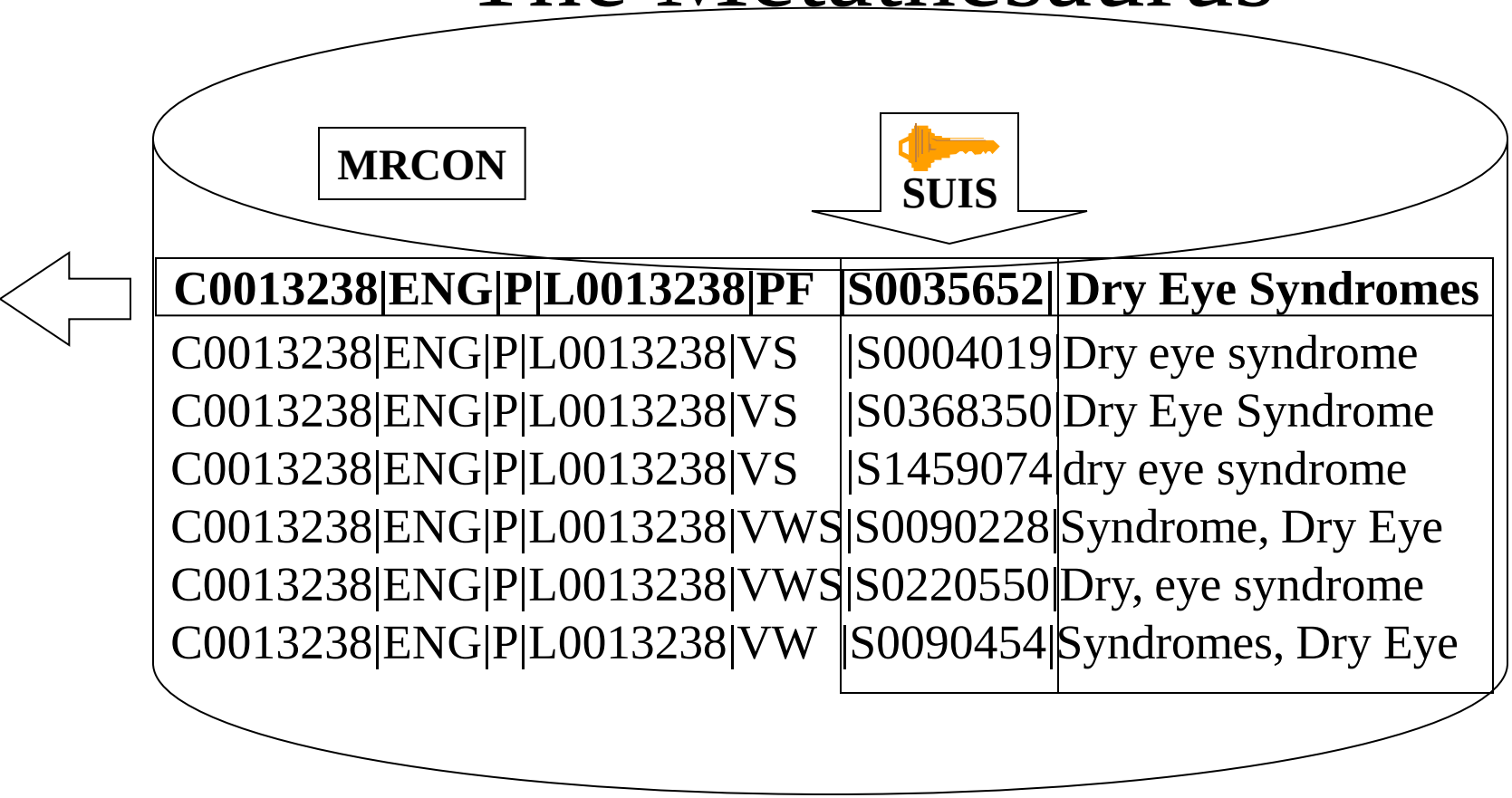


Query → **norm** → **Normed term** → **dry eye syndrome**

Using The Lexical Tools with The Metathesaurus



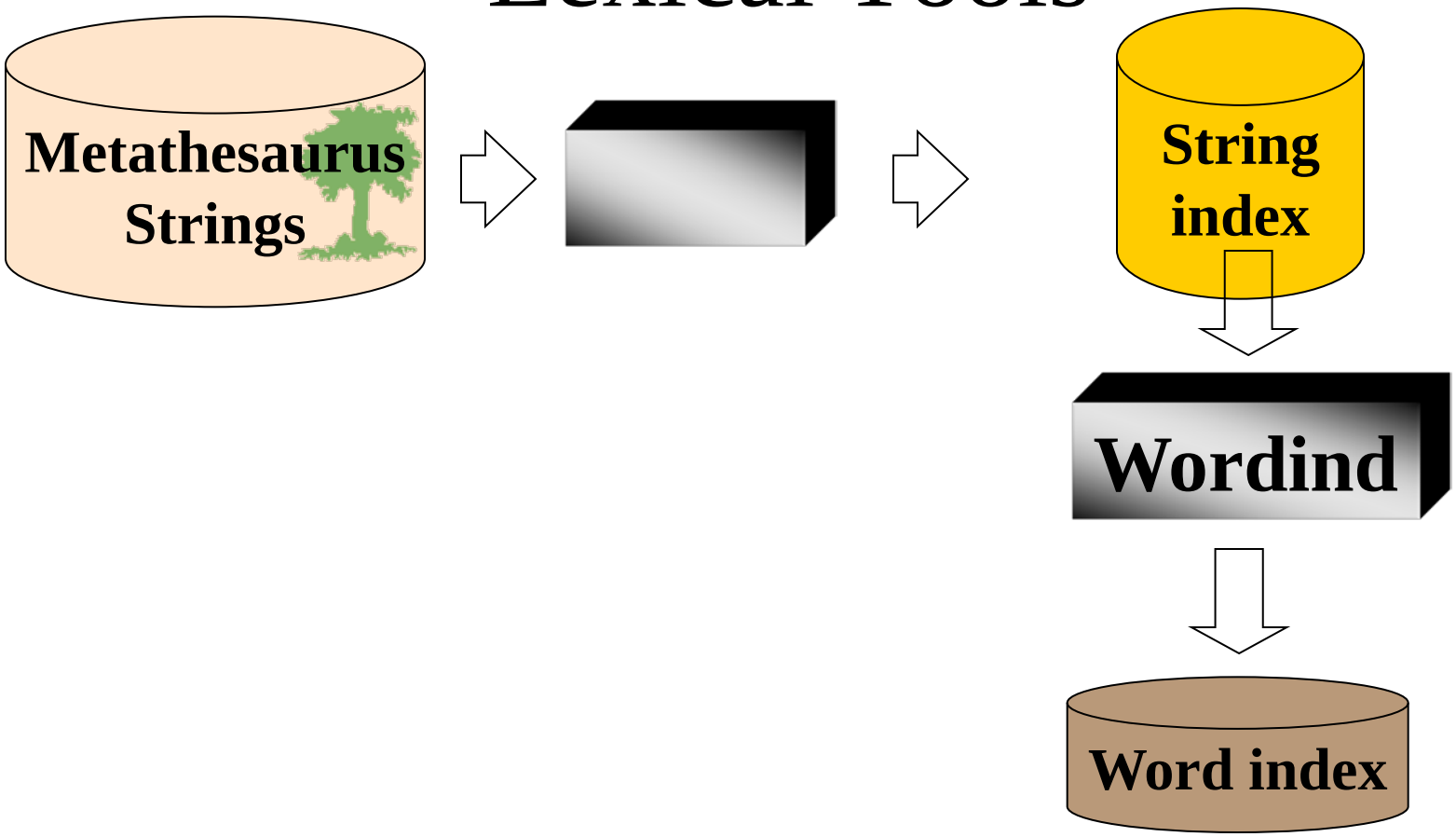
Using The Lexical Tools with The Metathesaurus



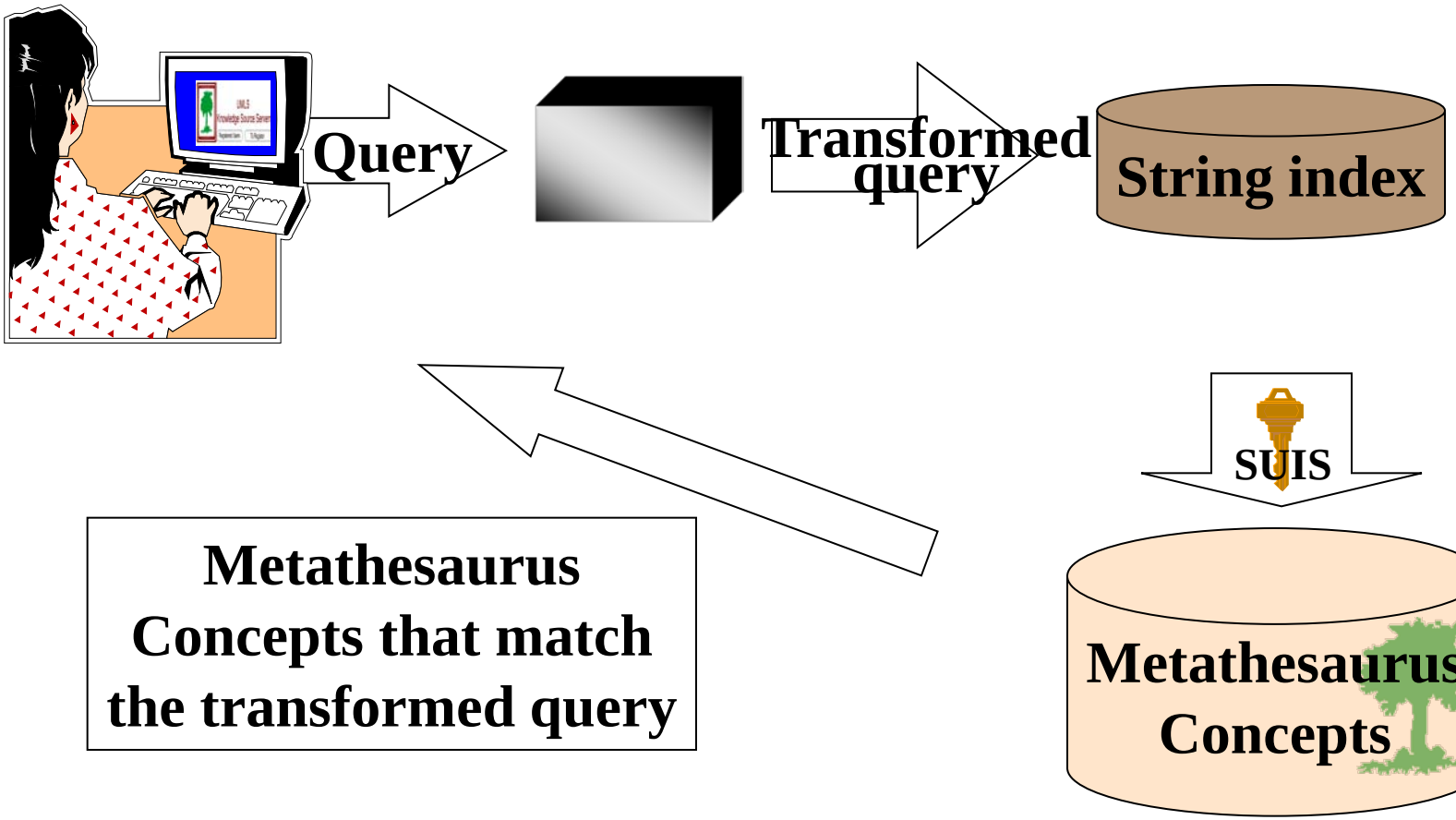
Building an Index Using The Lexical Tools

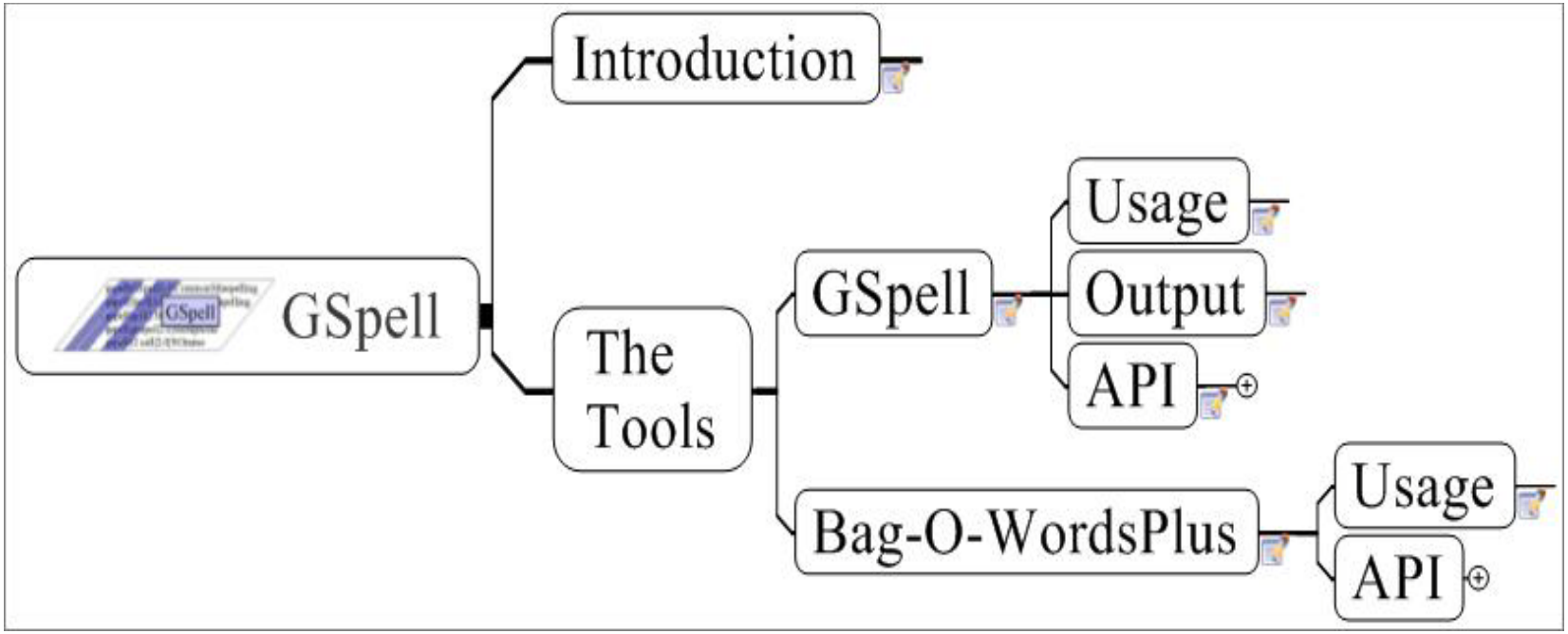
- Can we build a tool that increases precision?
  - Case
  - Constrain by part of speech
  - Filter to the lexicon
- Can we a tool that increases recall?
  - synonyms
  - derivations
  - acronyms and their expansions
  - spelling variants

Building an Index Using The Lexical Tools



Building an Index Using The Lexical Tools




GSpell
Introduction
The Tools
GSpell
Usage
Output
API
Bag-O-WordsPlus
Usage
API

Gspell: Introduction

- The GSpell program is a spelling suggestion tool that uses a mix of algorithms to retrieve close neighbors. This application is best suited to applications that index at the word or term level of tokenization.
- BagOWordsPlus is a phrase retrieval tool. This tool is useful to retrieve closest matching phrases to data such as strings from the Metathesaurus.

GSpell: Usage

Usage

GSpellFind.[*sh*|*bat*]

--**dictionary**=*NameOfDictionary*

[--**inputFile**=*Source*] [--**outputFile**=*target*]

[--**truncate**=*N*] [--**considerNCandidates**=*N*]

[--**maxEditDistance**=*N*]

GSpell: Indexing

Usage

GSpellIndex.[*sh*|*bat*]

--dictionary=*NameOfDictionary*

--inputFile=*SourceFile*

[--reportTime] [--version][--help]

- Format for the input file
  - One word per line

GSpell: Output

Input Term	Suggestion	Edit Distance	Rank	Method	Message

anonomous|**anonymous**|1.0|0.87|NGrams|
anonomous|**allonomous**|2.0|0.58|NGrams|
anonomous|**autonomous**|2.0|0.58|NGrams|
anonomous|**anadromous**|3.0|0.29|NGrams|
anonomous|**analogous**|3.0|0.29|NGrams|
anonomous|**anomalous**|3.0|0.29|NGrams|
anonomous|**anonymously**|3.0|0.29|NGrams|
anonomous|**anonymes**|3.0|0.29|Metaphone|
anonomous|**anonyms**|3.0|0.29|Metaphone|
anonomous|**acoprous**|4.0|0.11|NGrams|

GSpell: API

```
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();
```

BagOWordsPlus: Usage

Usage

BagOWordsPlusFind.[*sh*|*bat*]

--**dictionary**=*NameOfDictionary*

[--**inputFile**=*Source*] [--**outputFile**=*target*]

[--**truncate**=*N*] [--**considerNCandidates**=*N*]

[--**maxEditDistance**=*N*]

BagOWordsPlus: Indexing

Usage

BagOWordsPlusIndex.[*sh*|*bat*]

--dictionary=*NameOfDictionary*

--inputFile=*SourceFile*

[--reportTime] [--version][--help]

- Format for the input file
  - One phrase per line

BagOWordsPlus: Output

Input Term	Suggestion	Edit Distance

sleep|sleep|0.0

sleep|S-sleep|2.0

sleep|S sleep|2.0

sleep|REM sleep|4.0

sleep|deep sleep|5.0

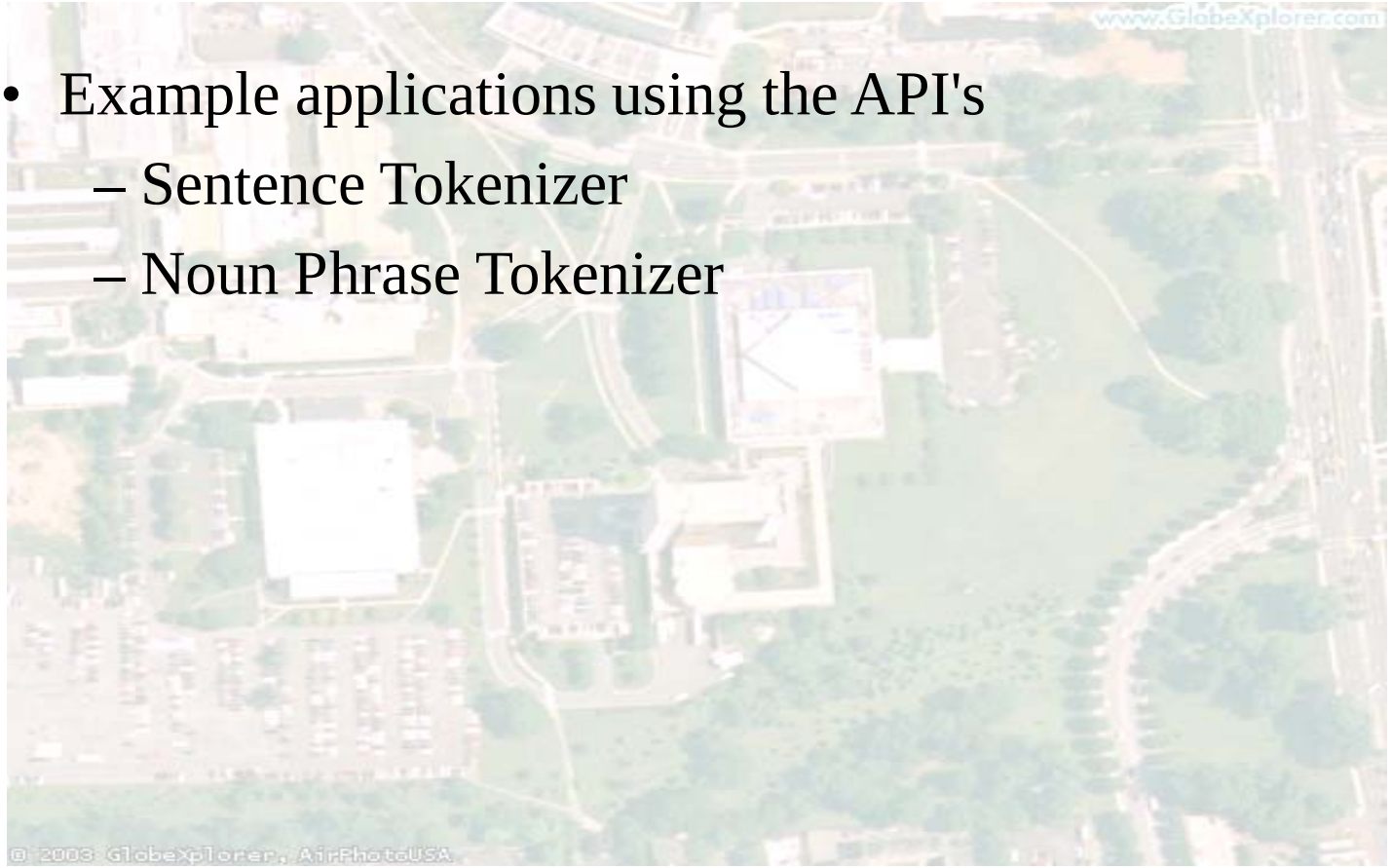
BagOWordsPlus: API

```
import gov.nih.nlm.nls.gspell.BagOWordsPlus;  // <-------These come from the gspell.jar
import gov.nih.nlm.nls.gspell.Candidate;

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

ir.cleanup();
```

SPECIALIST NLP Tools:
Table of Contents

- Logical/physical views of the functionalities
- The tools as stand-alone applications
  - Command line options
  - Example output
- API functionalities
  - The model of a document
    - Parts list
    - Structure
    - Details: Lexical Element
    - Details: Token

Table of Contents



- Example applications using the API's
  - Sentence Tokenizer
  - Noun Phrase Tokenizer

Introduction: Logical View

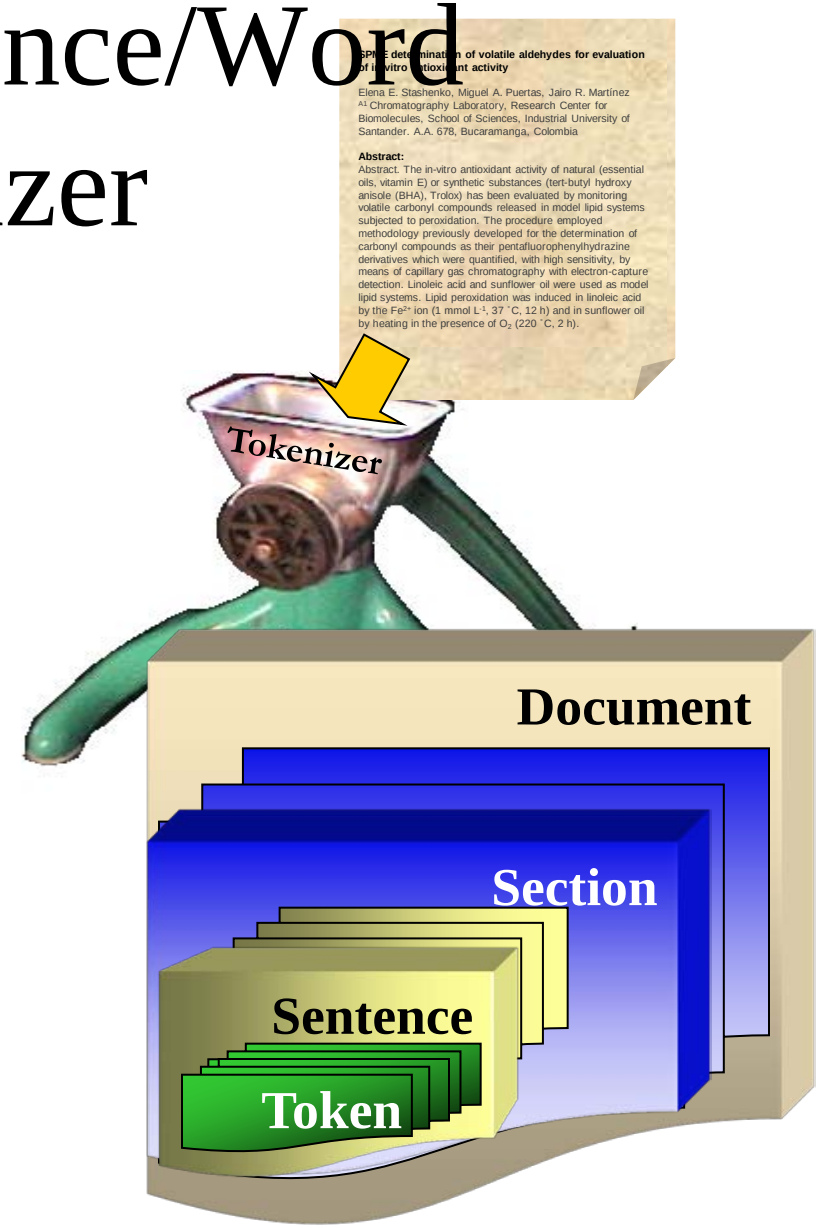
- Word Tokenizer
- Term Tokenizer
- Phrase Tokenizer
- Sentence Tokenizer
- Section Tokenizer

Introduction: Physical View

- Section/Sentence/Word Tokenizer
- Term Tokenizer
  - a.k.a lexical lookup, term recognizer
- Phrase Tokenizer
  - a.k.a phrase chunker, noun phrase extractor, parser

Section/Sentence/Word Tokenizer

- Tokenizes text into
  - –Sections (paragraphs)
    - –Sentences
      - –Tokens



SPECIALIST NLP Tools:

Tokenizer

Usage

tokenize.[*bat*|*sh*] [*Options*]

--fileName=*fileName*

--outputFileName=*fileName*

--inputType=[*freeText*|*medlineCitations*]

--sections

--sentences

--tokens

--pipedOutput

--indicate_citation_end

SPECIALIST NLP Tools:

Tokenizer

```
tokenize.bat --inputFile=5.txt --inputType=freeText --sentences --tokens
    --pipedOutput
```

Sentence|1|97|182|But those follow-up tests have been inconclusive, state and federal officials said.

Token|16|97|99|0|0|**But**|||

Token|17|101|105|1|0|**those**|||

Token|18|108|113|2|0|**follow**|||

Token|19|114|114|2|0|**-**|||

Token|20|115|116|3|0|**up**|||

Token|21|118|122|4|0|**tests**|||

Token|22|124|127|5|0|**have**|||

Token|23|129|132|6|0|**been**|||

Token|24|134|145|7|0|**inconclusive**|||

SPECIALIST NLP Tools:

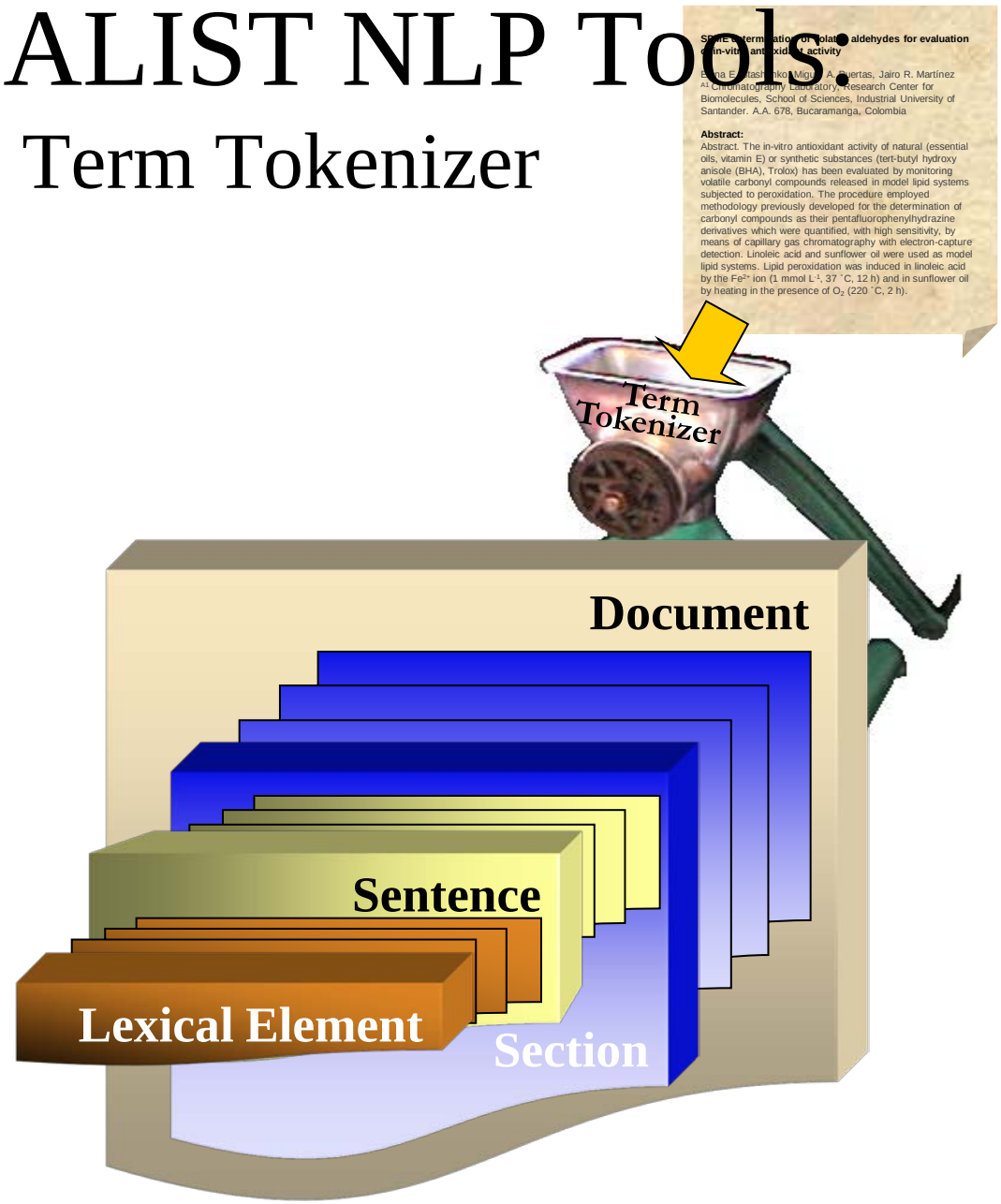
Tokenizer

```
// ==============+ Create a TokenizeAPI object +==
TokenizeAPI tokenizer = new TokenizeAPI( argv );
// ========================+ Tokenize the file +==
Document aDocument =
            tokenizer.processDocument( aFile);
Vector      tokens = aDocument.getTokens() ;
int numberOfTokens = tokens.size();
Token       aToken = null;
// ====================+ Print the tokens out +==
for ( int i = 0; i < numberOfTokens; i++ ) {
  aToken = (Token) tokens.get(i);
  System.out.println( aToken.toPipedString() );
}
```

SPECIALIST NLP Tools.

Term Tokenizer

- Chunks tokens into terms
  - From SPECIALIST Lexicon
  - From regular expressions



SPECIALIST NLP Tools:

Term Tokenizer

Usage

LexicalLookup.[*bat*|*sh*] [*Options*]

--**fileName**=*fileName*

--**outputFileName**=*fileName*

--**inputType**=[*freeText*|*HTML*|*medlineCitations*]

--**sections**

--**sentences**

--**lexicalElements**

--**lexicalEntries**

--**tokens**

--**pipedOutput**

SPECIALIST NLP Tools:

Term Tokenizer

```
LexicalLookup.bat --inputFile=5.txt --inputType=freeText
                  --lexicalElements --lexicalEntries --pipedOutput
```

Lexical Element|12|SHAPE:Unlabeled|*unknown*|**Richmond**|67|74

Lexical Element|13|LEXICON|prep|**for**|76|78

Lexical Element|14|LEXICON|adj|**further**|80|86

Lexical Element|15|LEXICON|verb|**testing**|88|94

Lexical Element |16|PUNCTUATION|punctuation|**.**|95|95

Lexical Element |17|LEXICON|prep|**But**|97|99

Lexical Element |18|LEXICON|det|**those**|101|105

Lexical Element |20|LEXICON|adj|**follow-up**|108|116

Lexical Element |23|LEXICON|noun|**tests**|118|122

Lexical Element |24|LEXICON|aux|**have**|124|127

SPECIALIST NLP Tools:

Term Tokenizer

```
LexicalLookup.bat --inputFile=5.txt --inputType=freeText
                  --lexicalElements --lexicalEntries --pipedOutput
```

Lexical Element|17|LEXICON|prep|**But**|97|99

LexicalEntry|but|conj|base|E0014465

LexicalEntry|but|prep|base|E0014464

Lexical Element|18|LEXICON|det|**those**|101|105

LexicalEntry|those|det|plural|E0060728

LexicalEntry|those|pron|base|E0060729

Lexical Element|20|LEXICON|adj|**follow-up**|108|116

LexicalEntry|follow-up|adj|base|E0028422

Lexical Element|23|LEXICON|noun|**tests**|118|122

LexicalEntry|tests|verb|pres3s|E0060349

LexicalEntry|tests|noun|plural|E0060348

SPECIALIST NLP Tools:

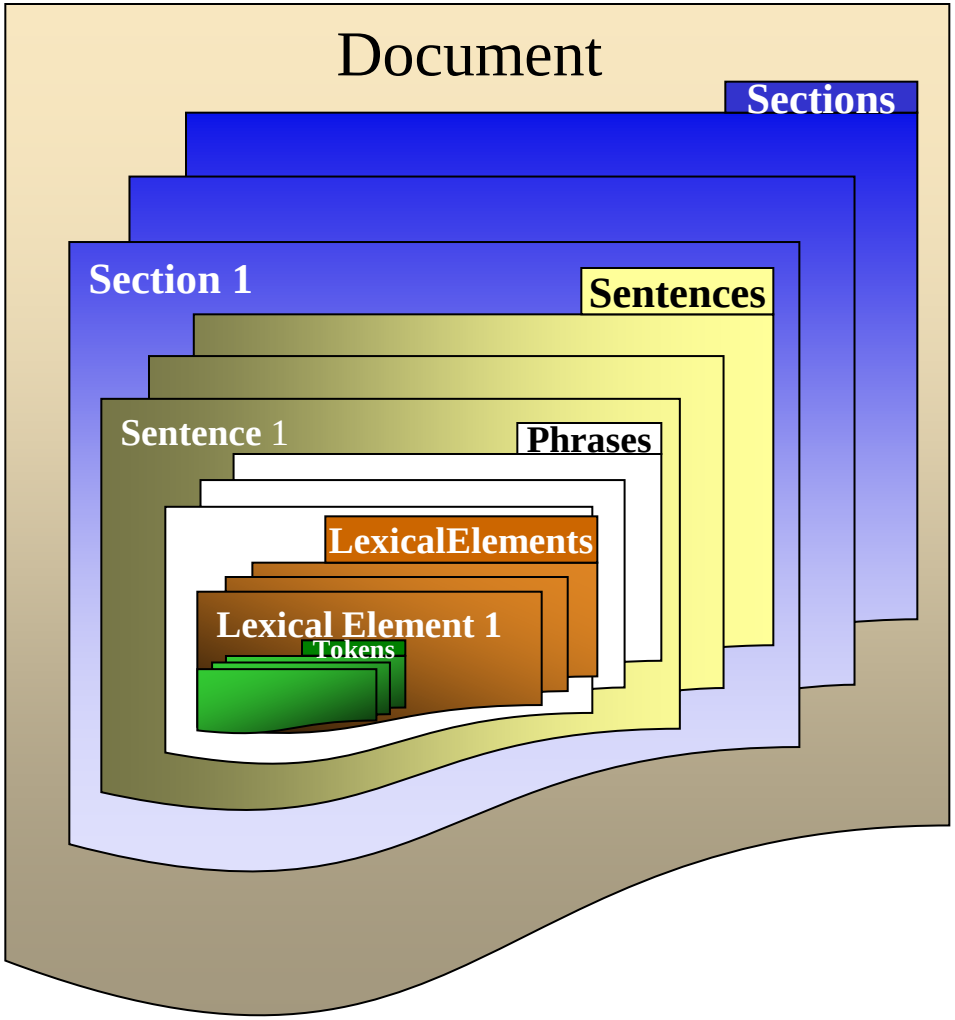
Term Tokenizer

```
// ==========+ Create a LexicalLookupAPI object +==
LexicalLookupAPI look = new LexicalLookupAPI(argv);
// ============================+ Chunk the file +==
Document aDocument = look.processDocument( aFile );

Vector les = aDocument.getLexicalElements();
int numberOfLexElements = les.size();
LexicalElement aLexElement = null;
// =============+ Print the LexicalElements out +==
for (int i = 0; i< numberOfLexElements; i++ ) {
  aLexElement = (LexicalElement) les.get(i);
  System.out.println(aLexElement.toPipedString());
}
```

SPECIALIST NLP Tools:

Phrase Tokenizer

- Chunks sentences into simple phrases



SPECIALIST NLP Tools:

Phrase Tokenizer

Usage

npParser.[*bat*|*sh*] [*Options*]

--**fileName**=*fileName*

--**outputFileName**=*fileName*

--**inputType**=[*freeText*|*HTML*|*medlineCitations*]

--**sections**

--**sentences**

--**phrases**|--**nps**|--**mincoMan**

--**lexicalElements**

--**lexicalEntries**

--**tokens**

--**pipedOutput**

SPECIALIST NLP Tools:

Phrase Tokenizer

```
npParser.bat --inputFile=5.txt --inputType=freeText --phrases
                --pipedOutput
```

Phrase|0|0|10|**The company**|*company*

Phrase|1|12|14|**has**|

Phrase|2|16|24|**forwarded**|

Phrase|3|26|39|**some materials**|*materials*

Phrase|4|41|62|**to a state laboratory**|*state laboratory*

Phrase|5|64|74|**in Richmond**|*Richmond*

Phrase|6|76|86|**for further**|*further*

Phrase|7|88|94|**testing**|

SPECIALIST NLP Tools:

Phrase Tokenizer

```
// ======================+ Create a Parser object +==
Parser parser = new Parser( argv );
// ============================+ Parse the file +==
Document aDocument = parser.processDocument(aFile);

Vector      phrases = aDocument.getPhrase() ;
Int numberOfPhrases = phrases.size();
Phrase      aPhrase = null;
// ======================+ Print the Phrases out +==
for ( int i = 0; i < numberOfPhrases; i++ ) {
  aPhrase = (Phrase) phrases.get(i);
  System.out.println( aPhrase.toPipedString() );
}
```

Document: A Model

How it's put together

Image from: **www.themodelcarmuseum.org/ 49box0.jpg**

Document Model: Parts List

- Sections
- Sentences
- Phrases
- Terms
- Words
- Lexicon Entries

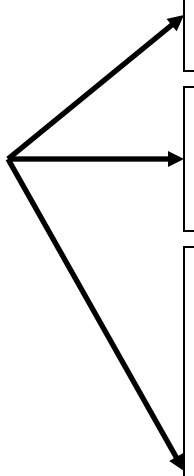
SPME determination of volatile aldehydes for evaluation of in-vitro antioxidant activity

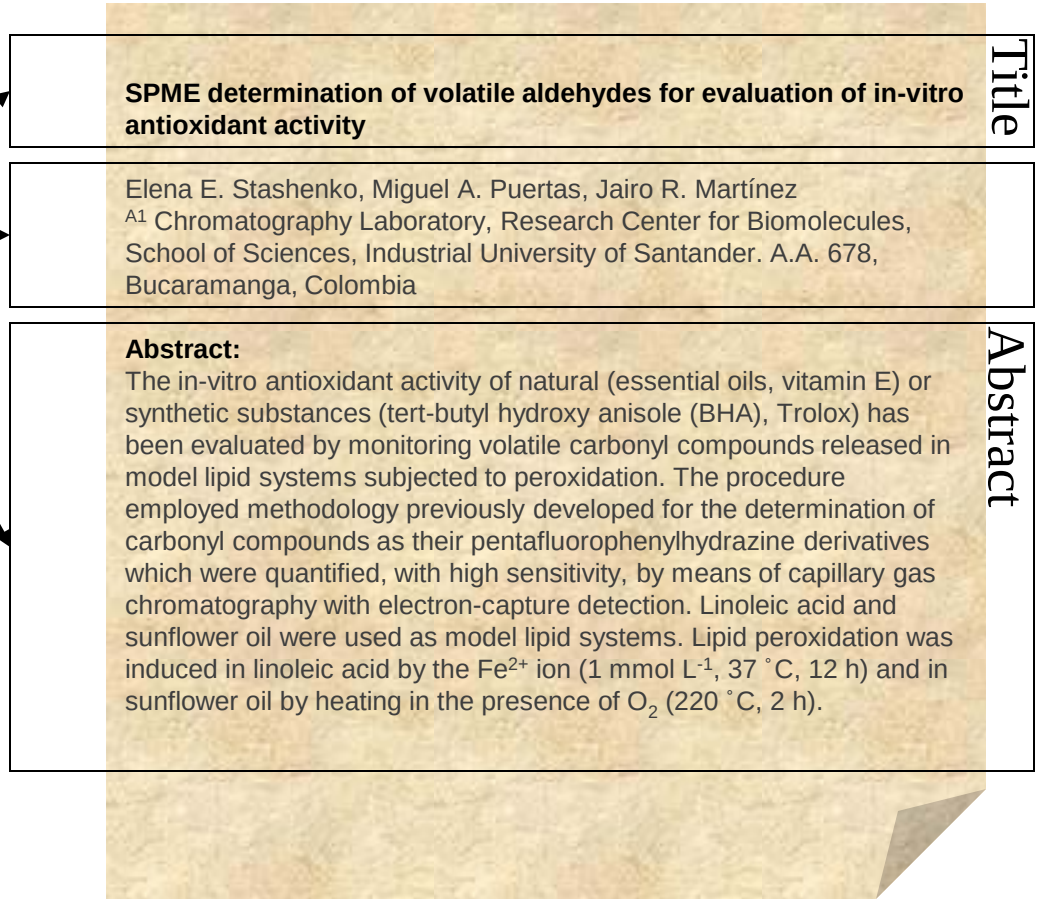
Elena E. Stashenko, Miguel A. Puertas, Jairo R. Martínez
[A1] Chromatography Laboratory, Research Center for Biomolecules, School of Sciences, Industrial University of Santander. A.A. 678, Bucaramanga, Colombia

Abstract:
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).

Document Model: Parts List

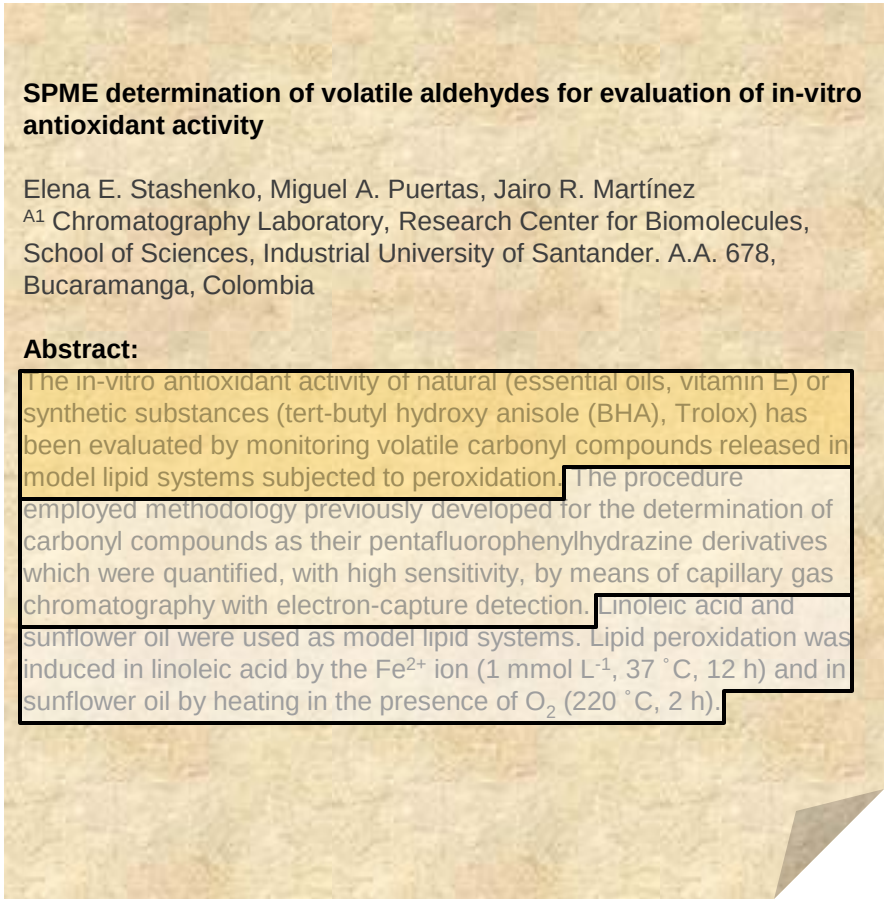
- Sections
- Sentences
- Phrases
- Terms
- Words
- Lexicon Entries





Document Model: Parts List

- Sections
- Sentences
- Phrases
- Terms
- Words
- Lexicon Entries



SPME determination of volatile aldehydes for evaluation of in-vitro antioxidant activity

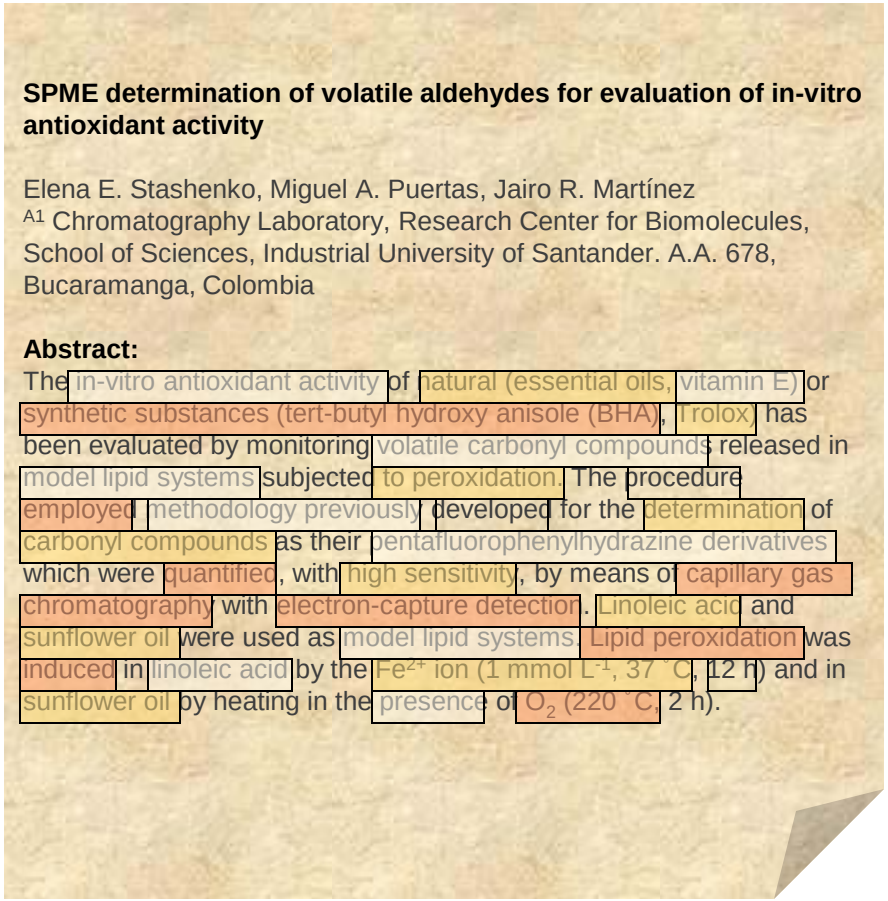
Elena E. Stashenko, Miguel A. Puertas, Jairo R. Martínez
[A1] Chromatography Laboratory, Research Center for Biomolecules, School of Sciences, Industrial University of Santander. A.A. 678, Bucaramanga, Colombia

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).

Document Model: Parts List

- Sections
- Sentences
- Phrases
- Terms
- Words
- Lexicon Entries



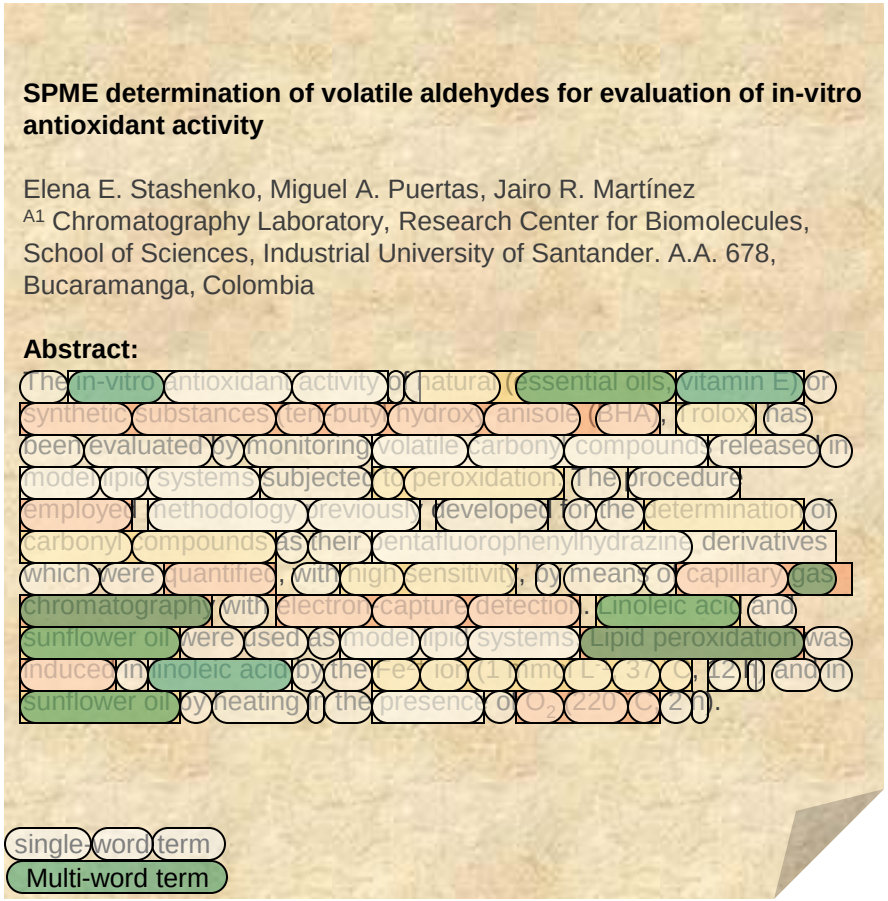
SPME determination of volatile aldehydes for evaluation of in-vitro antioxidant activity

Elena E. Stashenko, Miguel A. Puertas, Jairo R. Martínez
[A1] Chromatography Laboratory, Research Center for Biomolecules, School of Sciences, Industrial University of Santander. A.A. 678, Bucaramanga, Colombia

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).

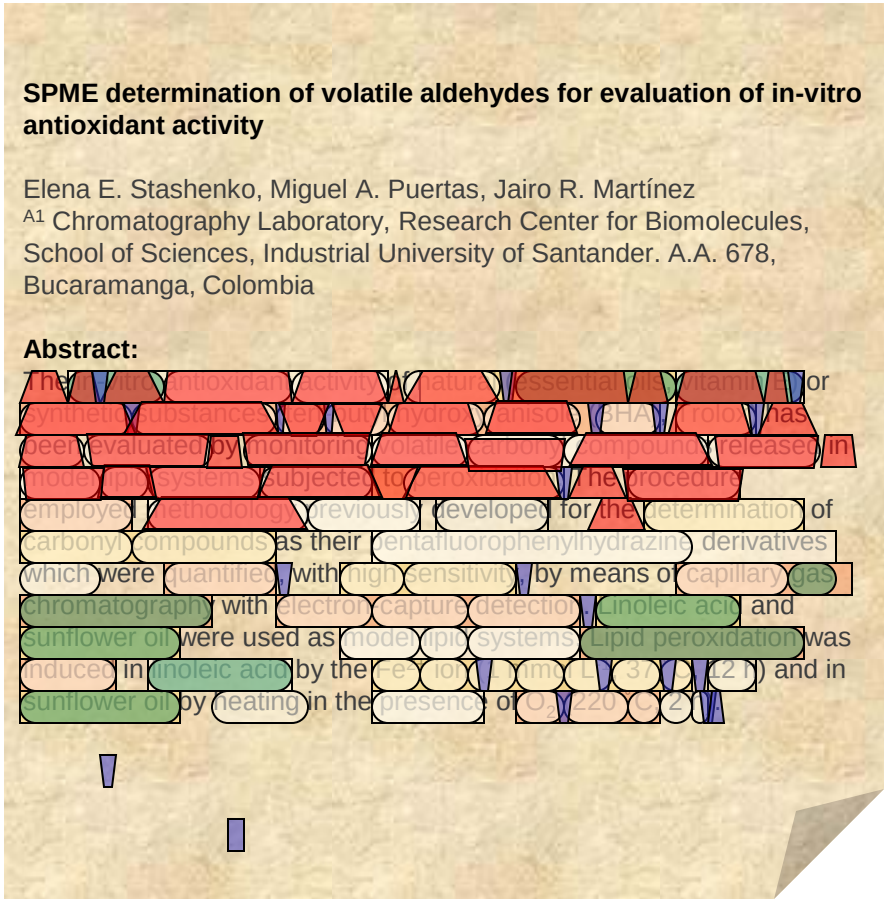
Document Model: Parts List

- Sections
- Sentences
- Phrases
- Terms
- Words
- Lexicon Entries



Document Model: Parts List

- Sections
- Sentences
- Phrases
- Terms
- Words
- Lexicon Entries



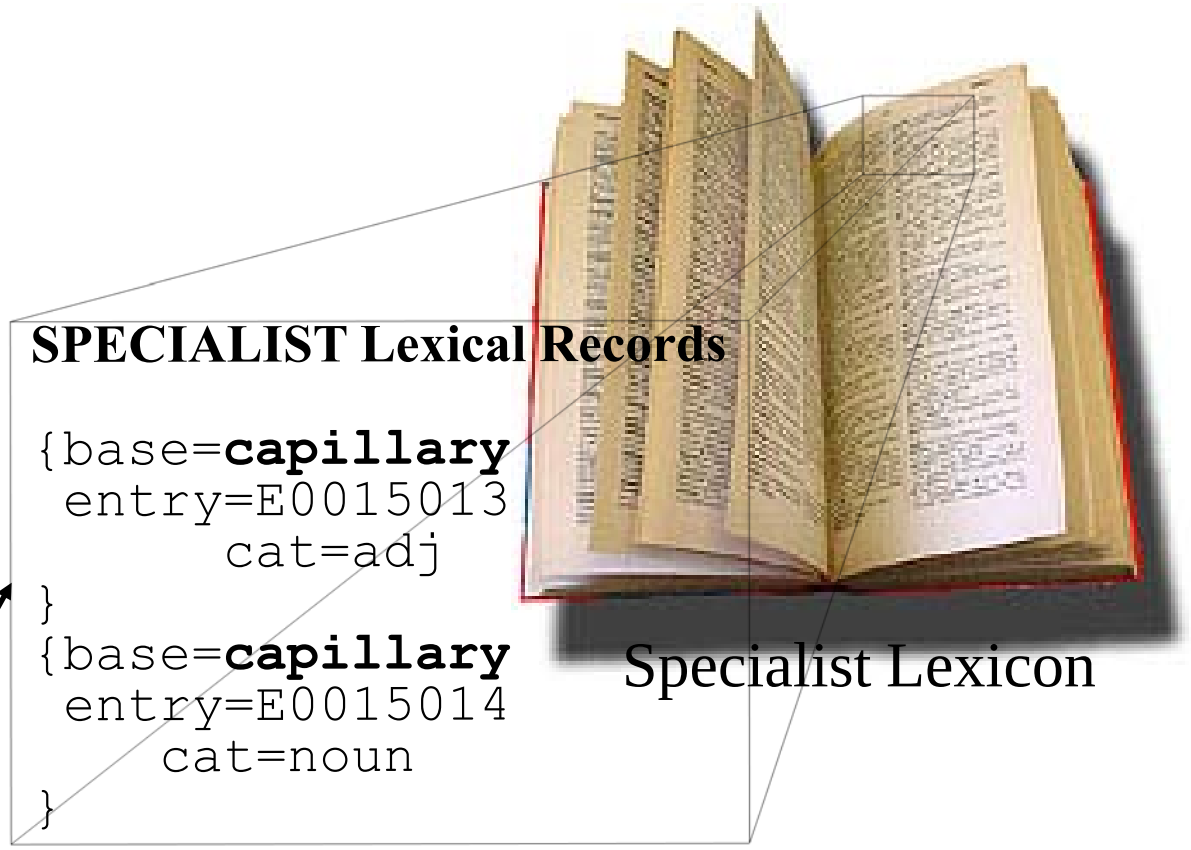
Document Model: Parts List

- Sections
- Sentences
- Phrases
- Terms
- Words
- Lexicon Entries

Specialist Lexicon

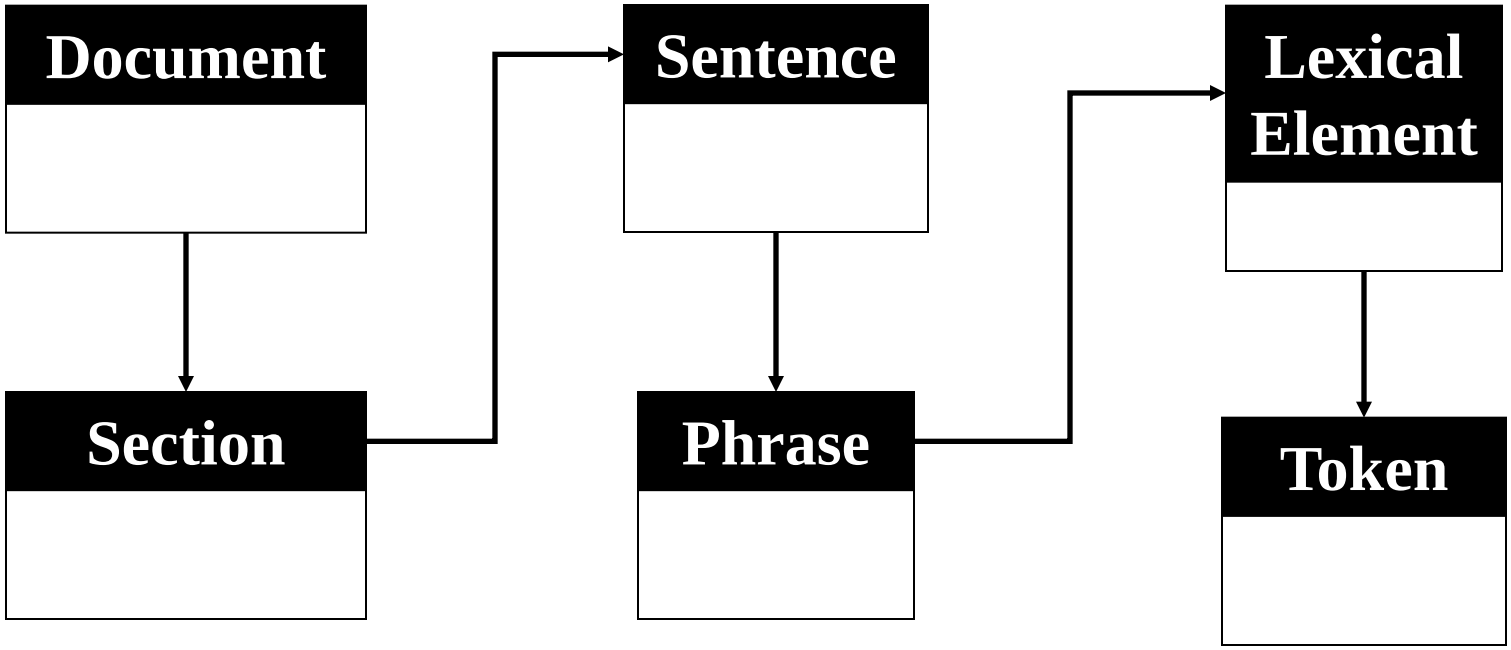
Document Model: Parts List

- Sections
- Sentences
- Phrases
- Terms
- Words
- Lexicon Entries



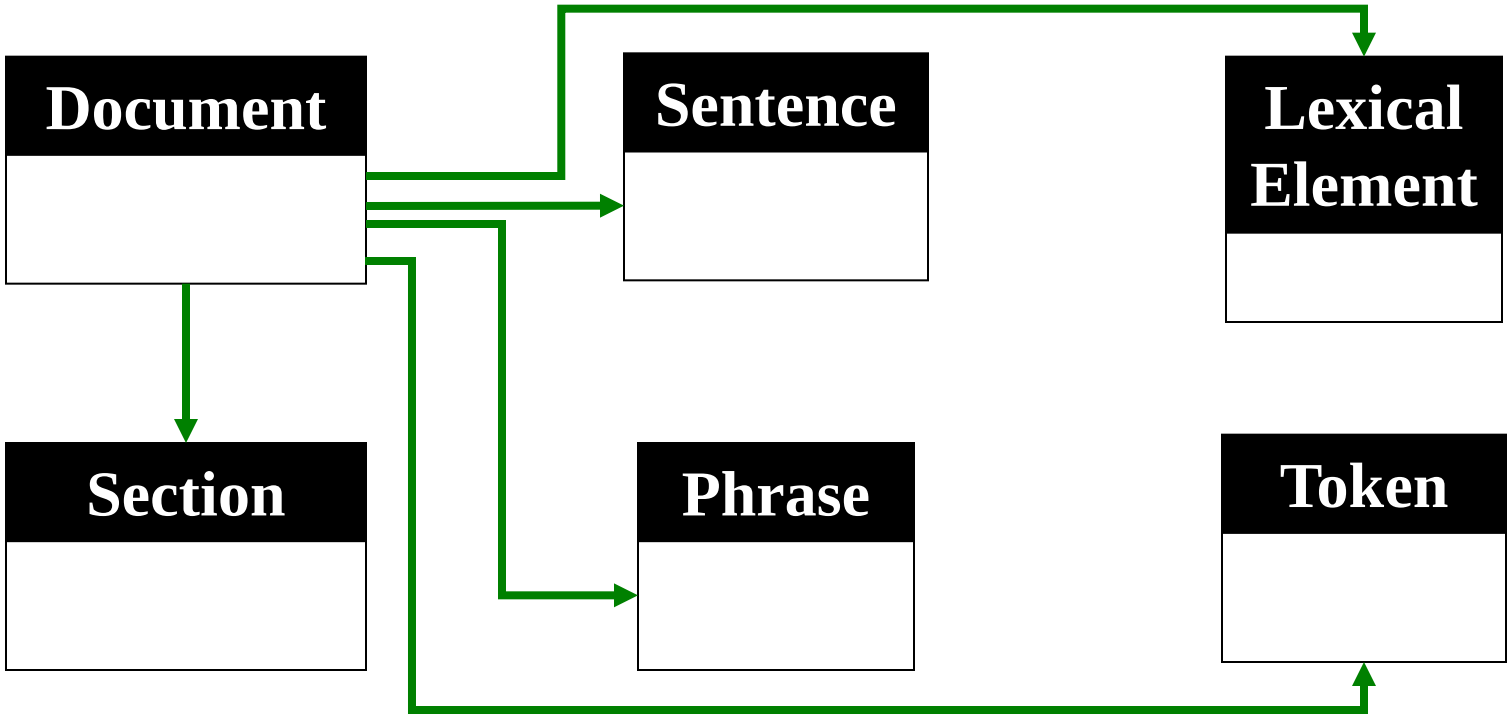
Document Model: Structure

Document Model: Structure



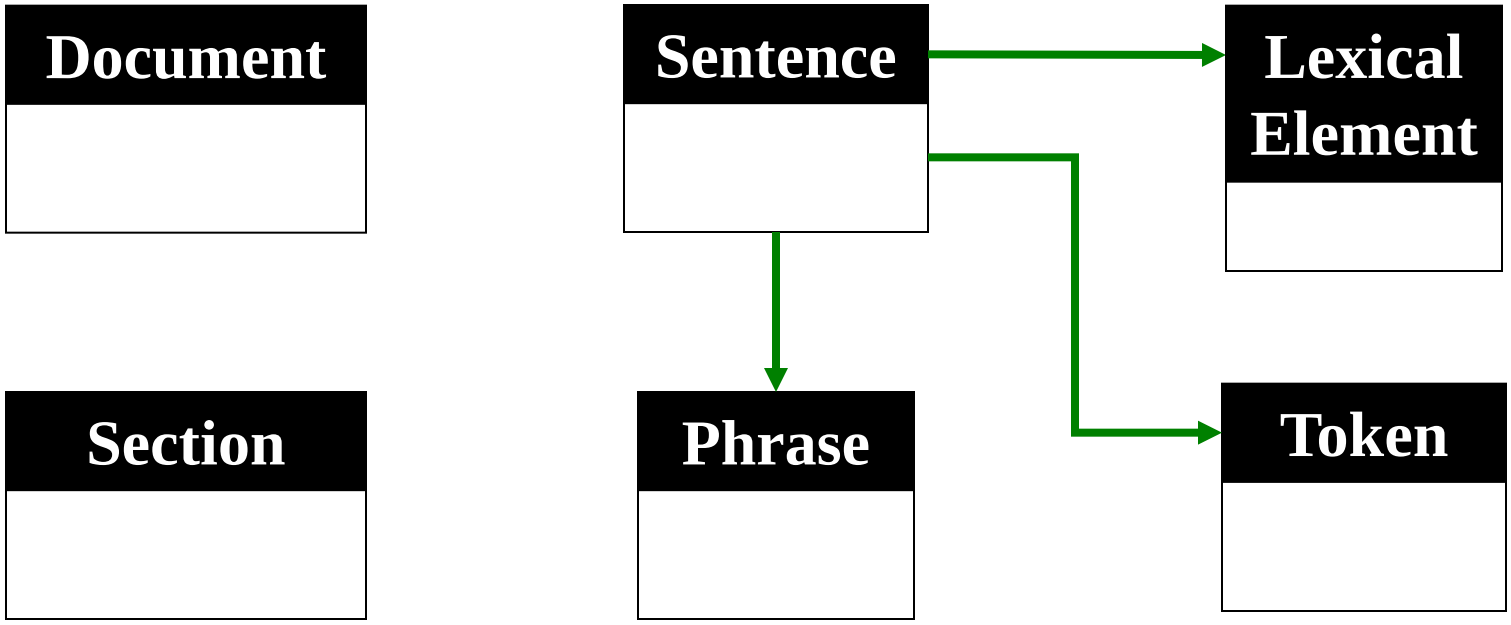
One to Many Relationship

Document Model: Structure



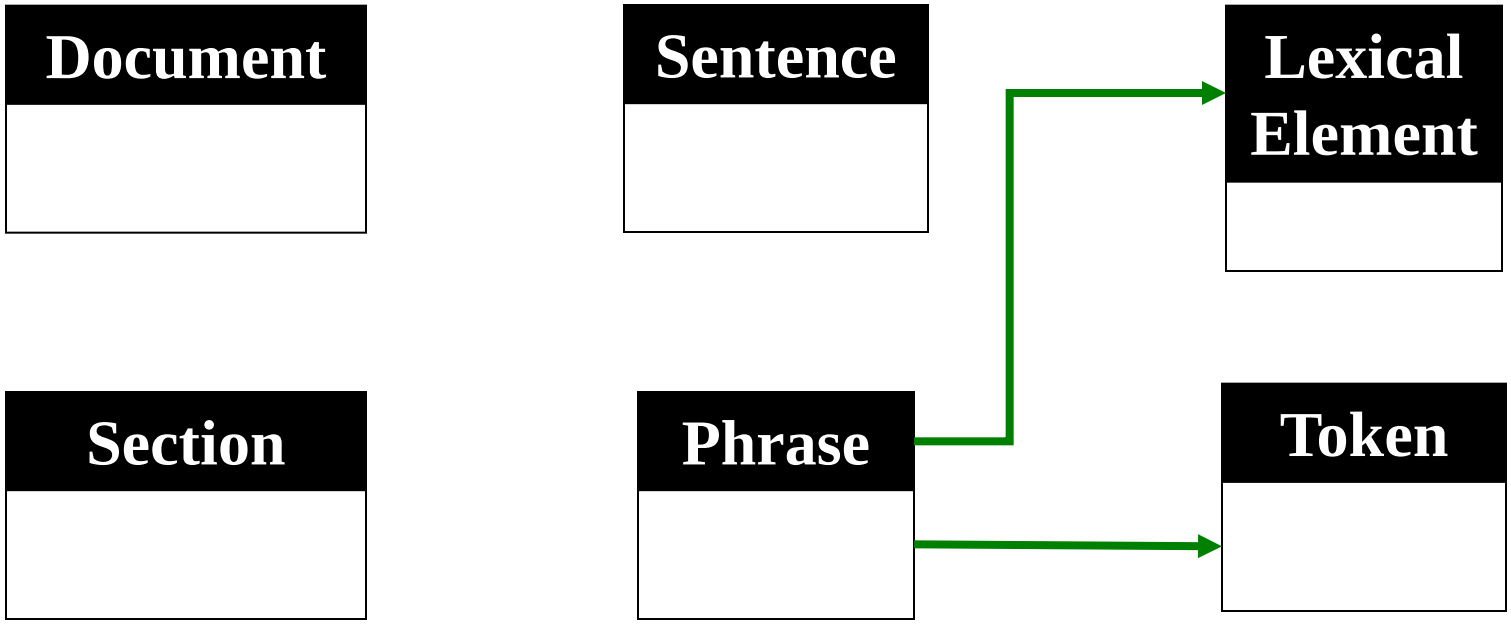
One to Many Relationship

Document Model: Structure



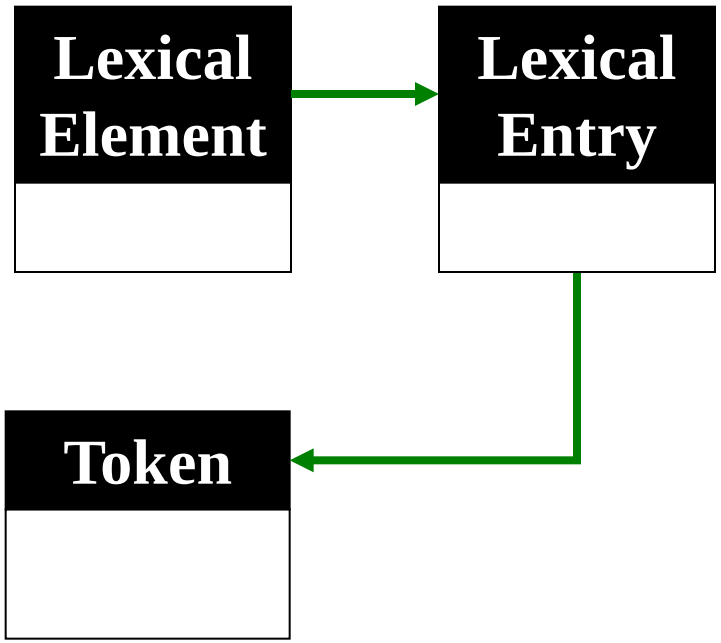
One to Many Relationship

Document Model: Structure



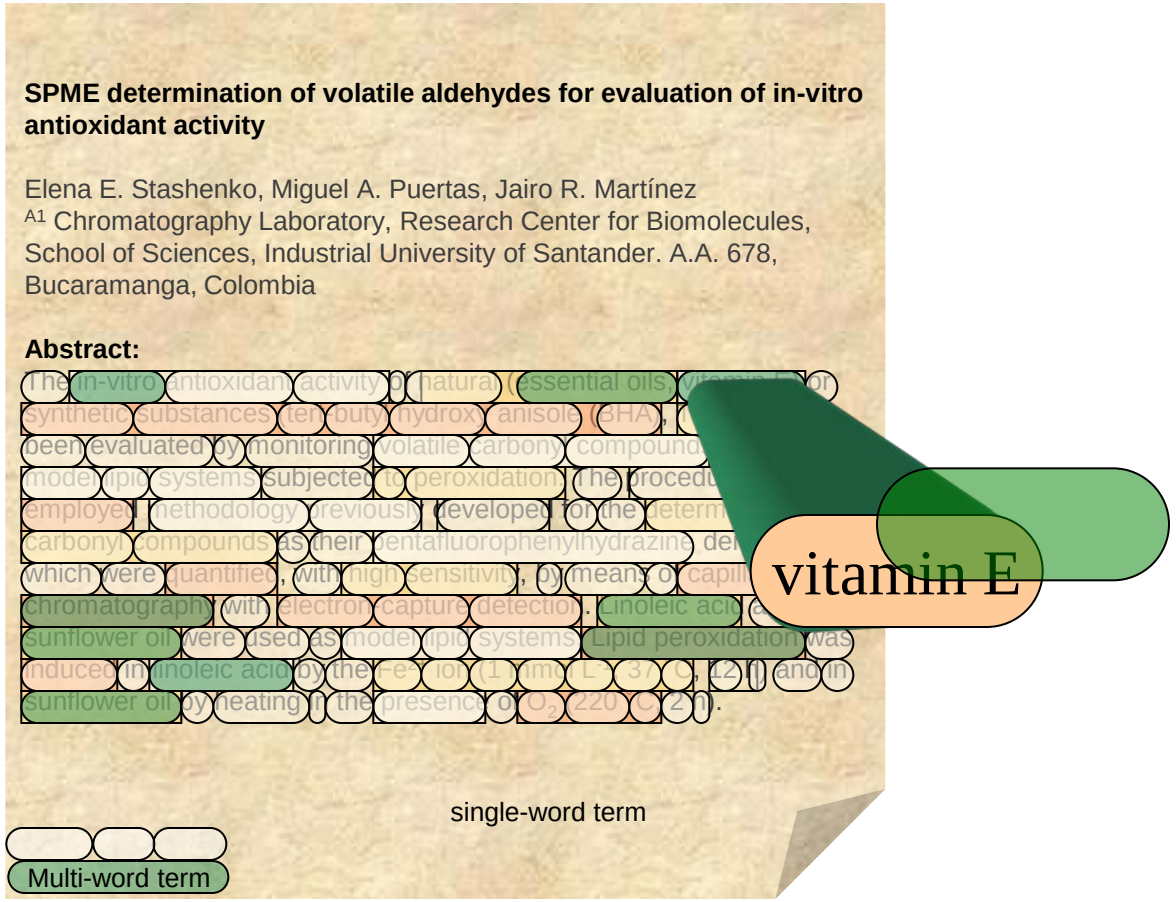
One to Many Relationship

Document Model: Structure



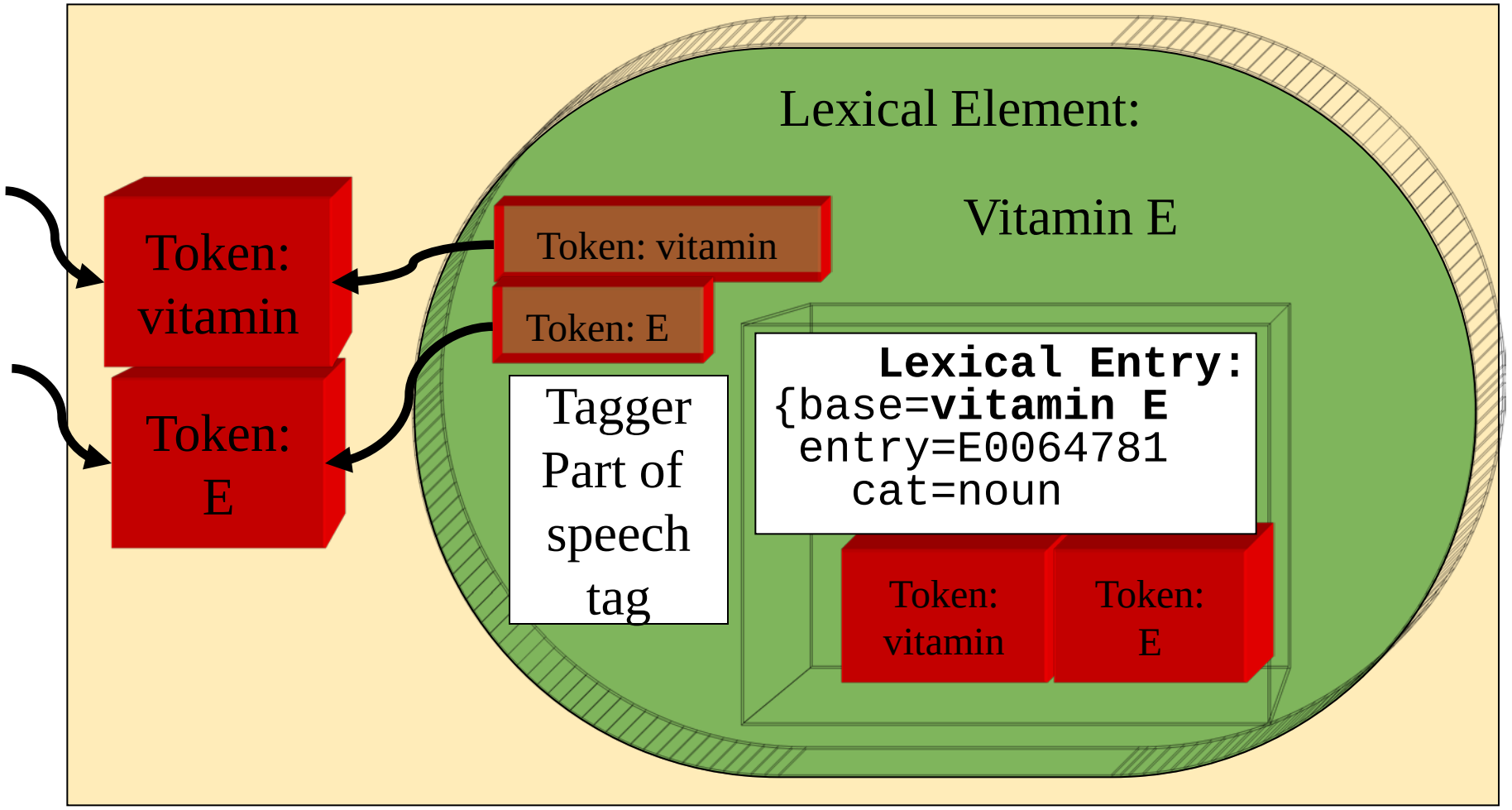
One to Many Relationship

Document Model:
Lexical Element

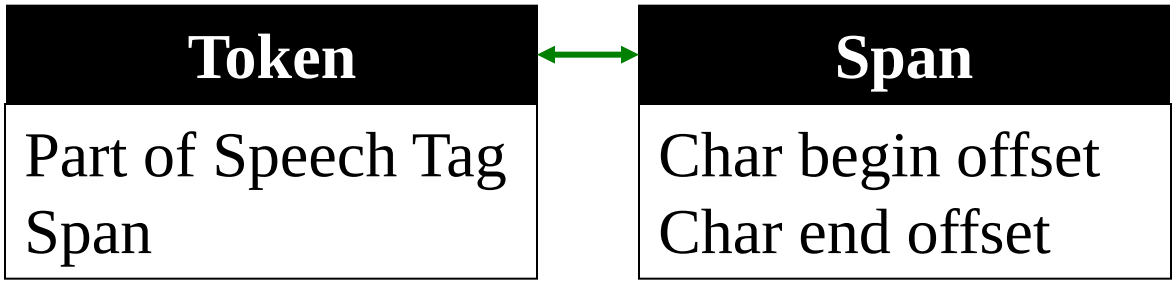


Document Model:

Lexical Element



Document Model:
Token

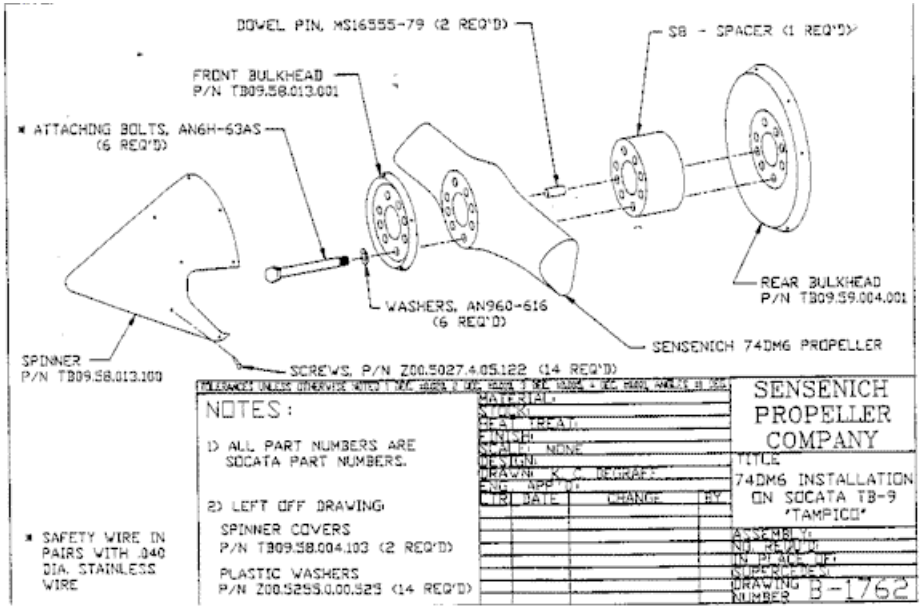


Document Model:
Phrase

Phrase
String displayTags() String displayVariants() List getAllVariants() UMLS_ConceptPointer getConceptPointer() UMLS_ConceptPointer[] getConcepts() List getDerivedPhrases() ArrayList getFinalMappings() List getLexicalElements() List getNp() String getNpString() List getNpTokens() String getOriginalString()

Phrase (cont.)
int getPhrasePosition() int getSizeOfPhrase() String getTrimmedString() boolean isOfPhrase() boolean isPrepPhrase() String toMincoManString() String toMoString() String toPipedString() String toString() String toSyntaxString()

Assembly Instructions

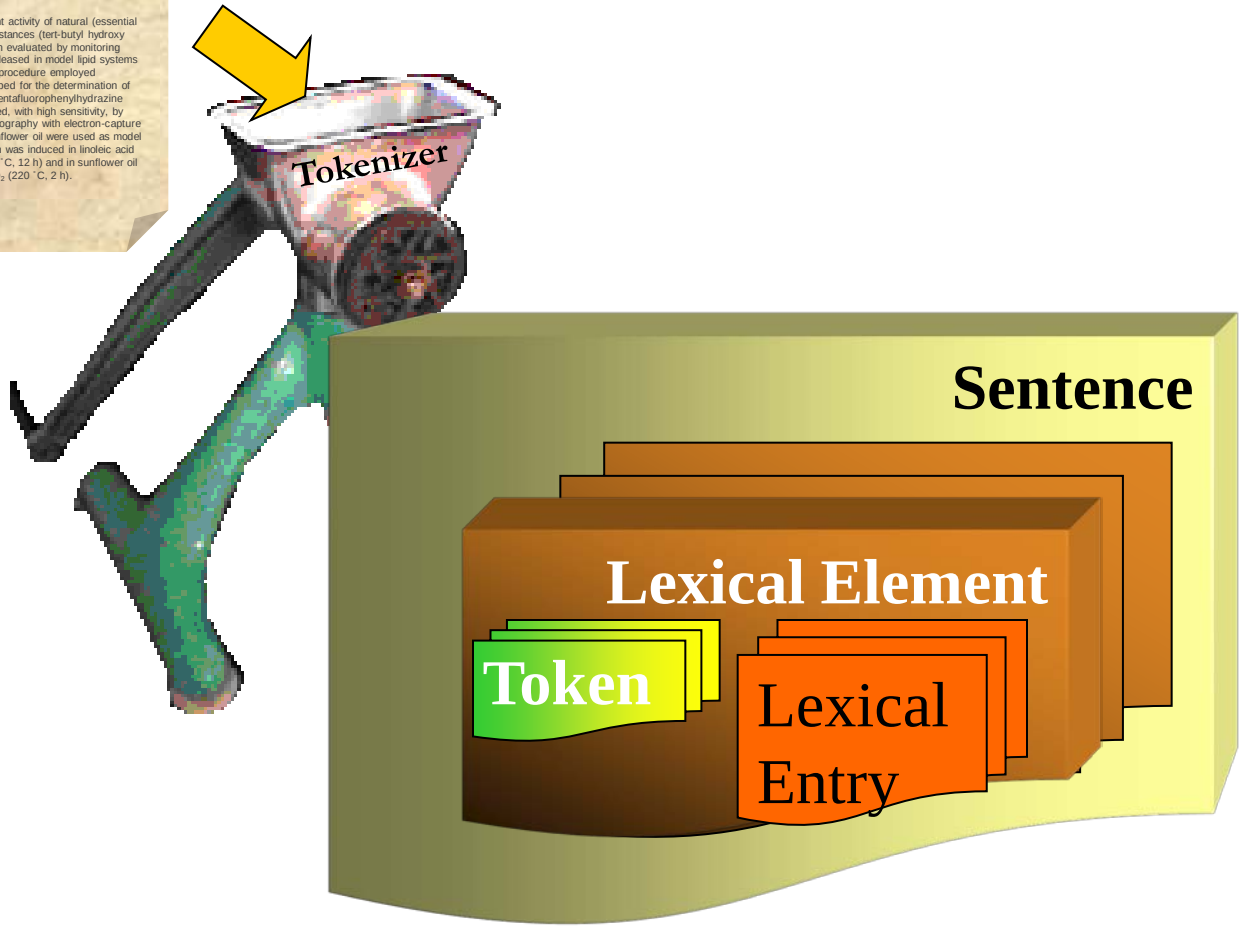


Term Tokenizer

SPME determination of volatile aldehydes for evaluation of in-vitro antioxidant activity

Elena E. Stashenko, Miguel A. Puertas, Jairo R. Martínez
[A1] Chromatography Laboratory, Research Center for Biomolecules, School of Sciences, Industrial University of Santander. A.A. 678, Bucaramanga, Colombia

Abstract:
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).



Phrase Tokenizer

SPME determination of volatile aldehydes for evaluation of in-vitro antioxidant activity

Elena E. Stashenko, Miguel A. Puertas, Jairo R. Martínez [A1] Chromatography Laboratory, Research Center for Biomolecules, School of Sciences, Industrial University of Santander. A.A. 678, Bucaramanga, Colombia

Abstract:
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).