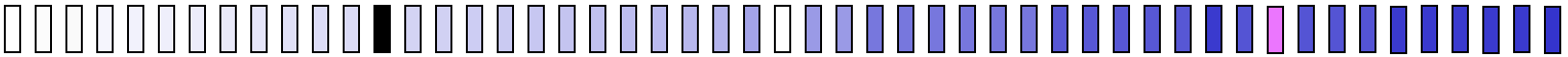
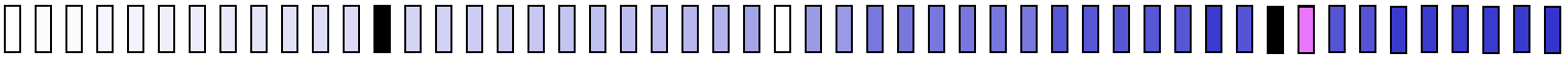
Additional NLS Tools

- Knowledge Source Server Java Client API
- NLS's Java NLP tools
- MMTx
- GSpell



Knowledge Source Server Java Client API

- XML over RMI
- Java UMLS Object Model



UMLS Knowledge Source Server (UMLSKS)

UMLSKS Version 2.1 UMLS Releases: 2002 2002AB

Metathesaurus Semantic Network SPECIALIST Lexicon

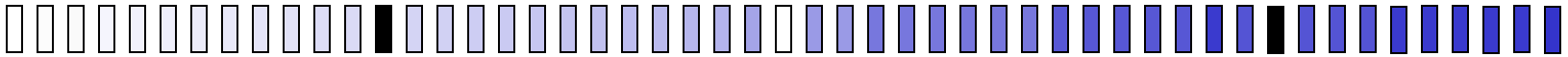
Documentation Resources Views/Profiles Logout

UMLSKS User's Guide

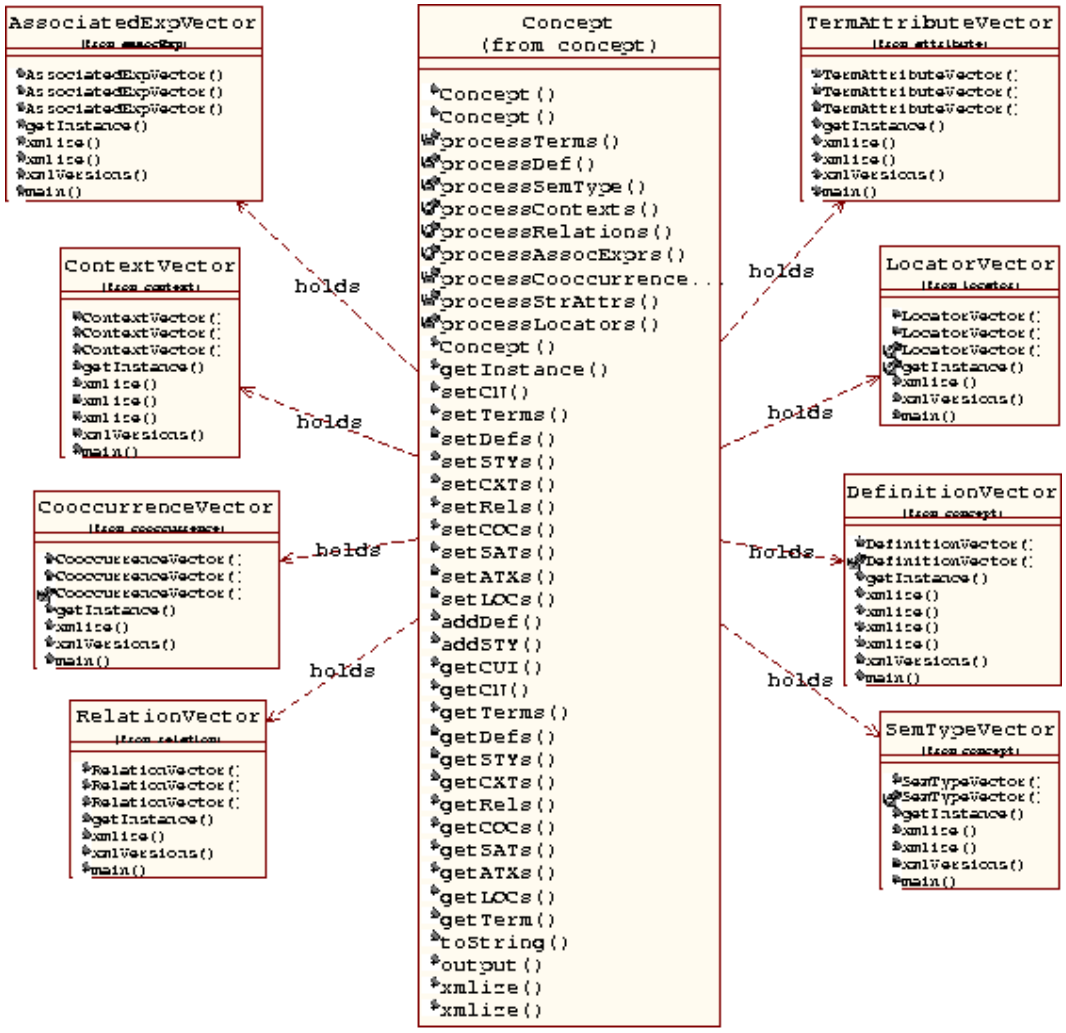
UMLSKS API Download and Documentation

Chapter 5. Building UMLSKS Software Applications

Chapter Contents:

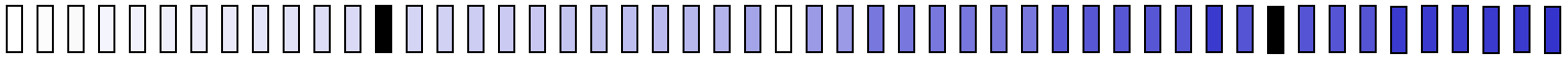


Knowledge Source Server Java



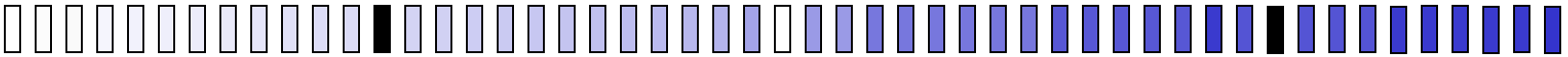
Knowledge Source Server Java Client API

```
// ======================+ Initialize the client +==
KSSRetrieverV2_1 retriever = (KSSRetrieverV2_1)
   Naming.lookup("//umlsks.nlm.nih.gov/KSSRetriever");
// ====================+ Send a request to client +==
char[] result = retriever.findBasicConcept(ksYear,
                                           termName,
                                               sabs,
                                           language,
                                KSSRetriever.NormalizeString,
                                              false );
// ====================+ Convert the XML into ... +==
ConceptVector concepts = ConceptVector.getInstance(
                                       String.valueOf(result))
```



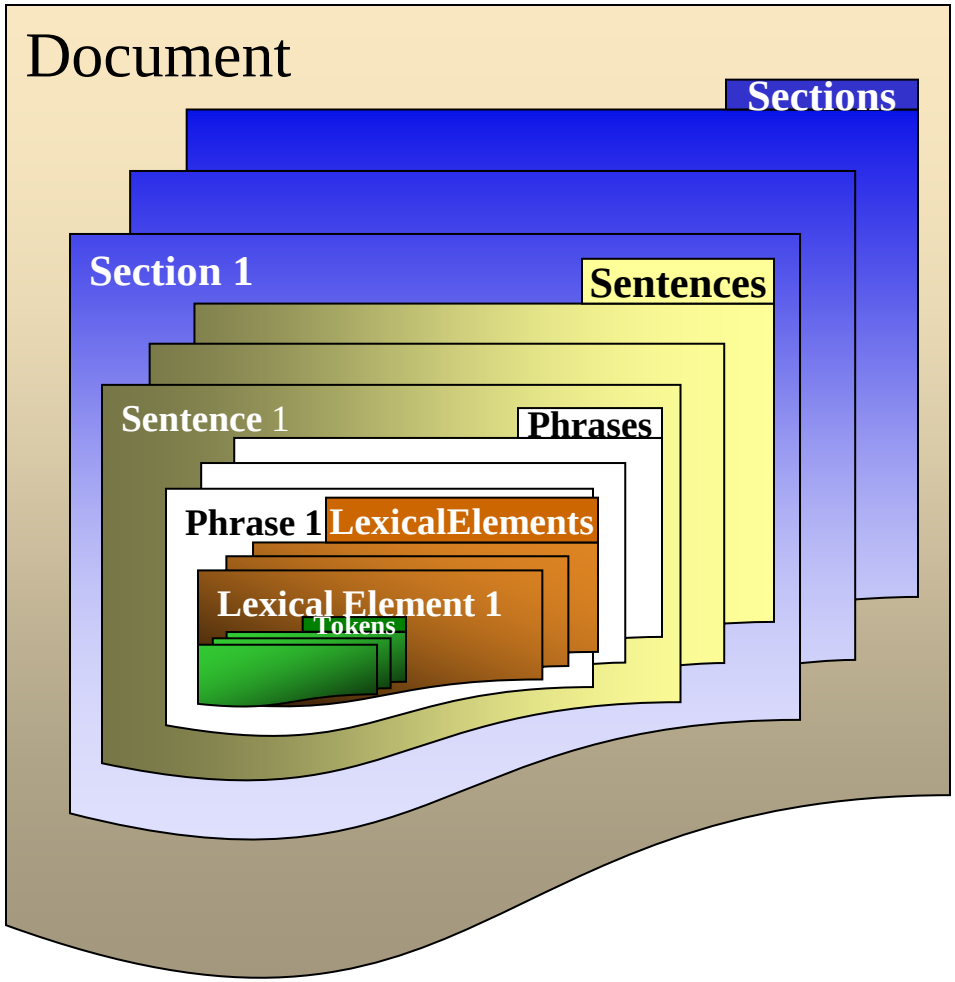
Knowledge Source Server Java Client API

```
<concept>
    <cui>C0032615</cui>
    <cn>Fatty Acids, Polyunsaturated</cn>
    <term>
      <lui>L0032615</lui>
      <tn>Fatty Acids, Polyunsaturated</tn>
      <ts>P</ts>
      <lat>ENG</lat>
      <termVariant>
        <sui>S0010240</sui>
        <stt>VW</stt>
        <str>Acids, Polyunsaturated Fatty</str>
        <strSource>
          <sab>MSH2002</sab><tty>PM</tty><scd>D005231</scd><srl>0</srl>
```



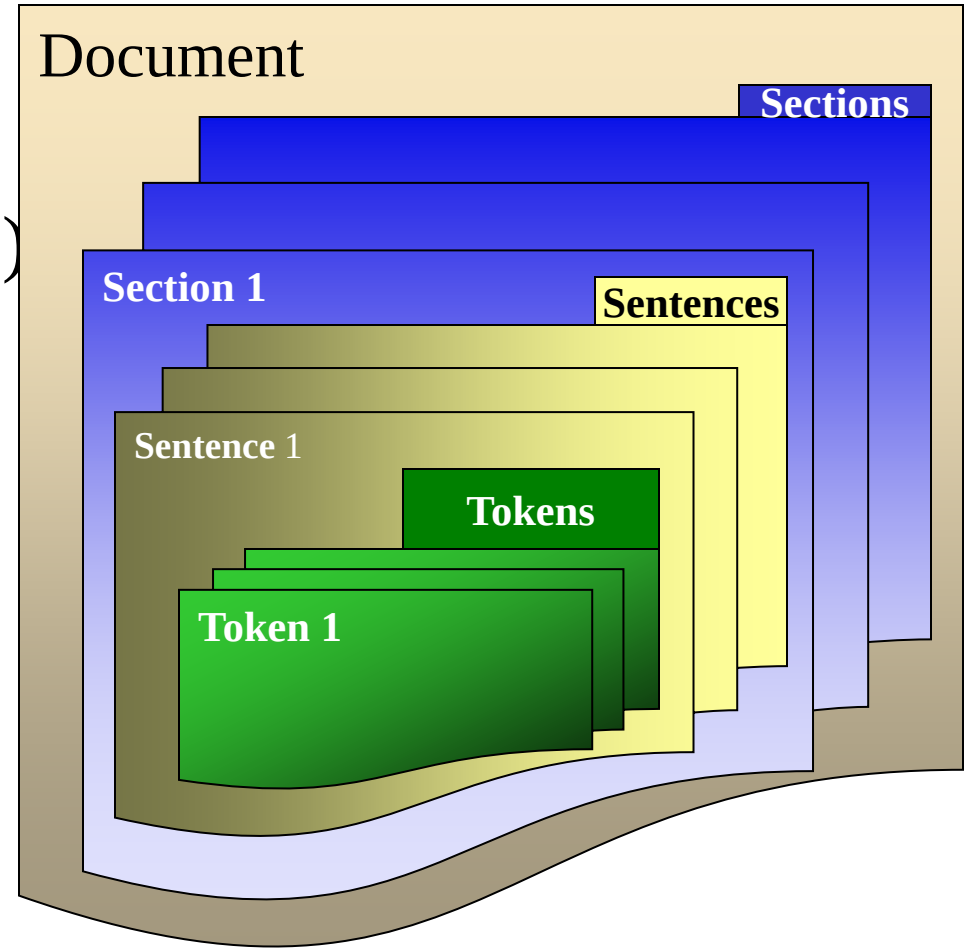
NLS Java NLP Tools

- Tokenizer
- Lexical Lookup
- NP Parser
  - Document Centric
  - Java Programs and API's



Java NLP Tools: Tokenizer

- Tokenizes text into
  - Sections (paragraphs)
  - Sentences
  - Tokens
- Can handle
  - FreeText
  - HTML
  - MedLINE Abstracts



Java NLP Tools: Tokenizer

Usage

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

--fileName=*fileName*

--outputFileName=*fileName*

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

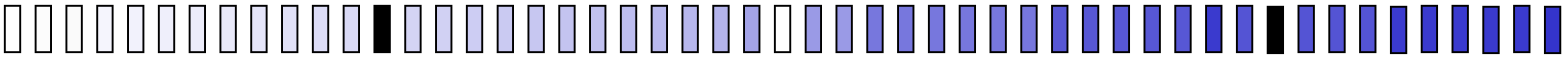
--sections

--sentences

--tokens

--pipedOutput

--indicate_citation_end

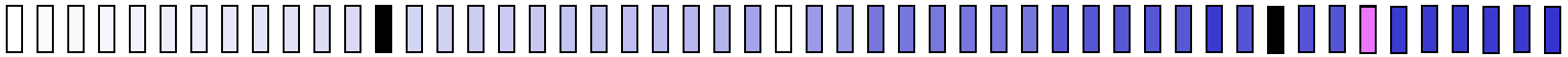


Java NLP Tools: Tokenizer

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

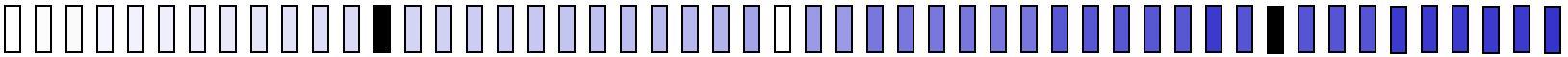
Sentence|1|97|182|But those follow-up tests have been inconclusive, state and federal officials said.

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**|||



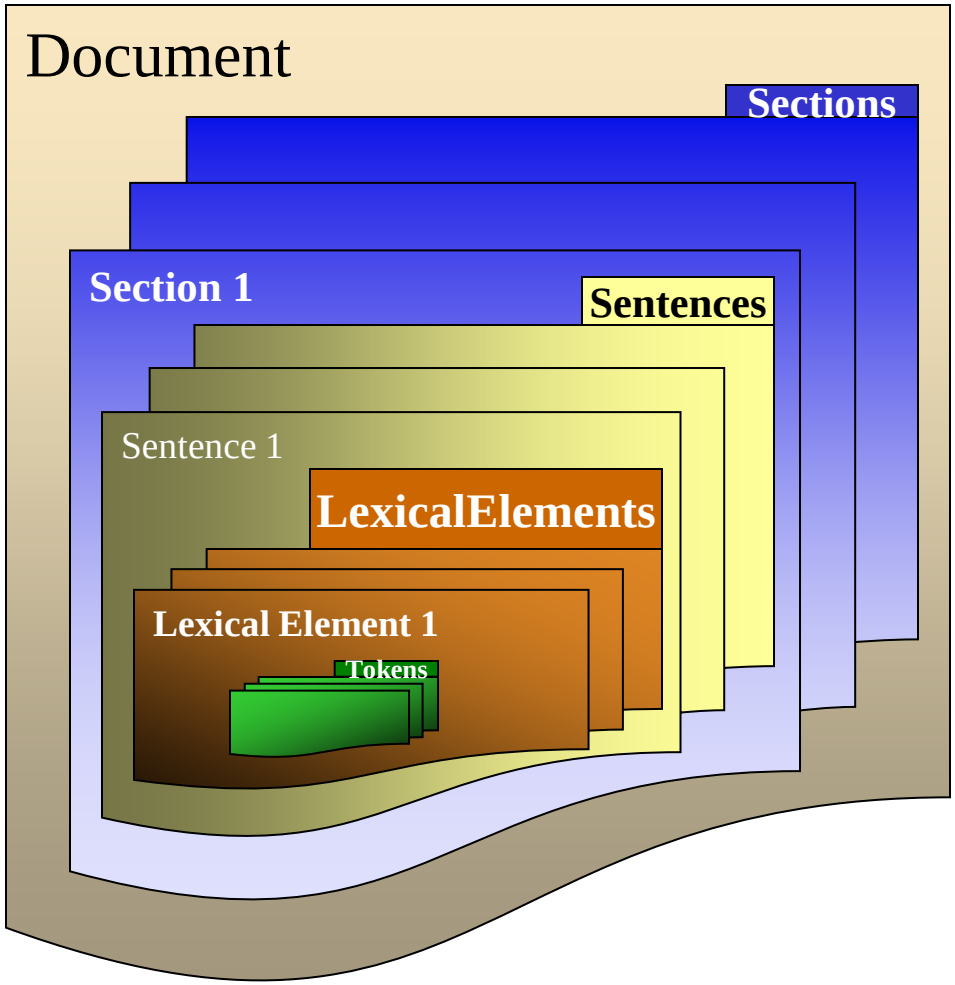
Java NLP Tools: Tokenizer

```
// ==============+ 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() );
}
```



NLP Tools: Lexical Lookup

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



Java NLP Tools: Lexical Lookup

Usage

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

--fileName=*fileName*

--outputFileName=*fileName*

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

--sections

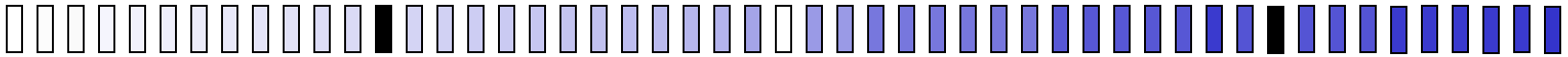
--sentences

--lexicalElements

--lexicalEntries

--tokens

--pipedOutput



Java NLP Tools: Lexical Lookup

```
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

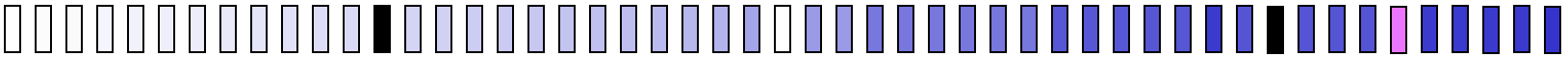
Lexical Element|20|LEXICON|adj|**follow-up**|108|116

LexicalEntry|follow-up|adj|base|E0028422

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

LexicalEntry|tests|verb|pres3s|E0060349

LexicalEntry|tests|noun|plural|E0060348



Java NLP Tools: Lexical Lookup

```
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

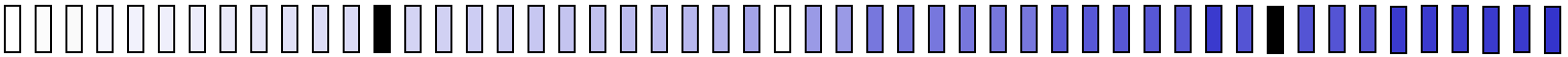
Lexical Element |17|LEXICON|prep|**But**|97|99

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

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

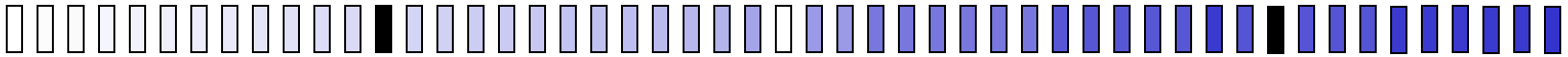
Lexical Element |23|LEXICON|noun|**tests**|118|122

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



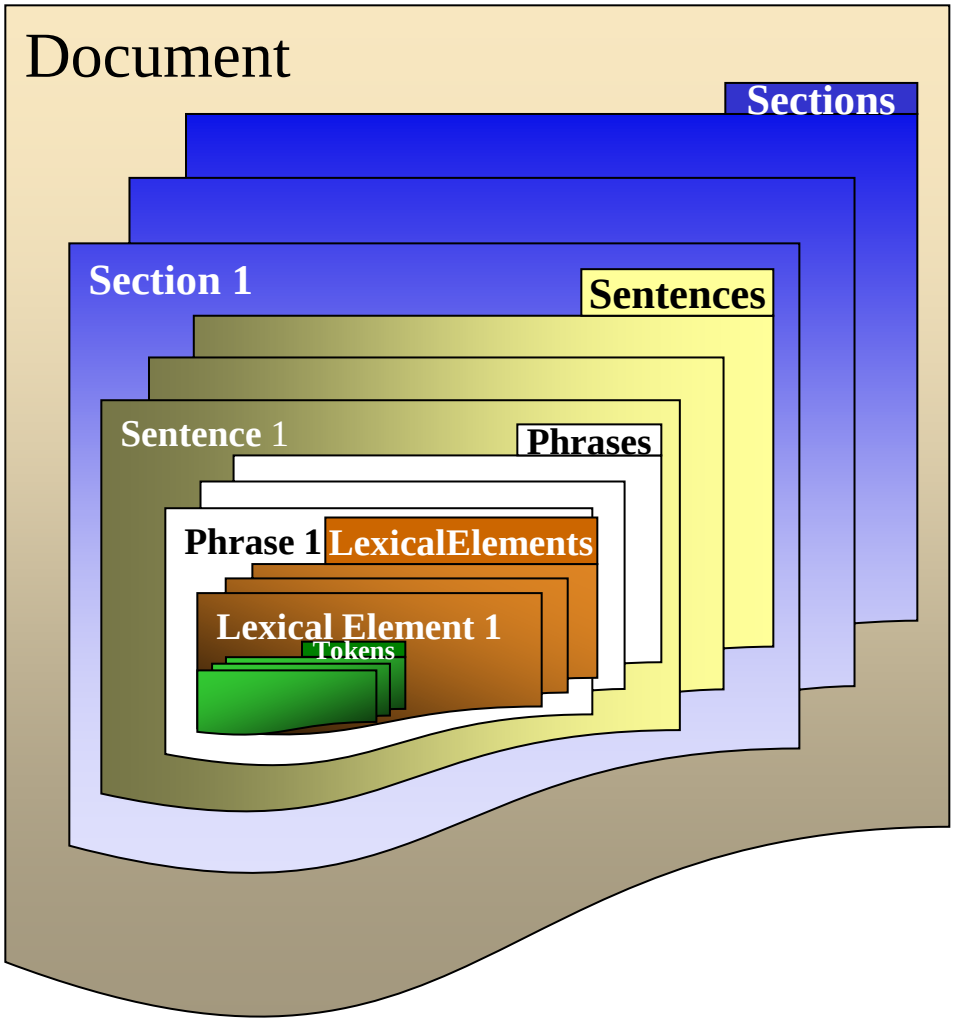
Java NLP Tools: Lexical Lookup

```
// ==========+ 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());
}
```



NLP Tools: NpParser

- Chunks sentences into simple phrases



Java NLP Tools: NpParser

Usage

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

--fileName=*fileName*

--outputFileName=*fileName*

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

--sections

--sentences

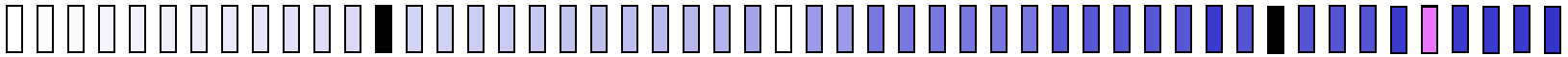
--phrases|**--nps**|**--mincoMan**

--lexicalElements

--lexicalEntries

--tokens

--pipedOutput



Java NLP Tools: NpParser

```
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**|

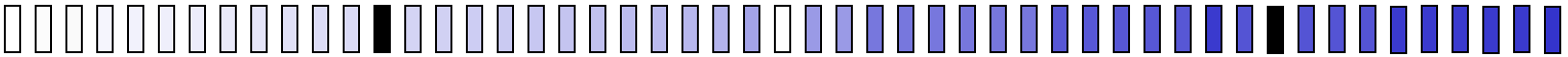
Phrase|3|26|39|**some materials**|*materials*

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

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

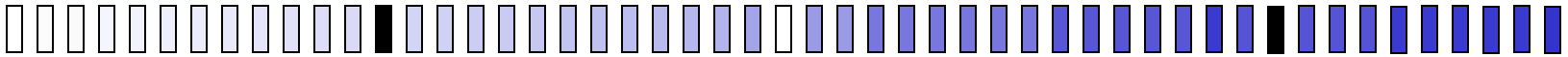
Phrase|6|76|86|**for further**|*further*

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



Java NLP Tools: NpParser

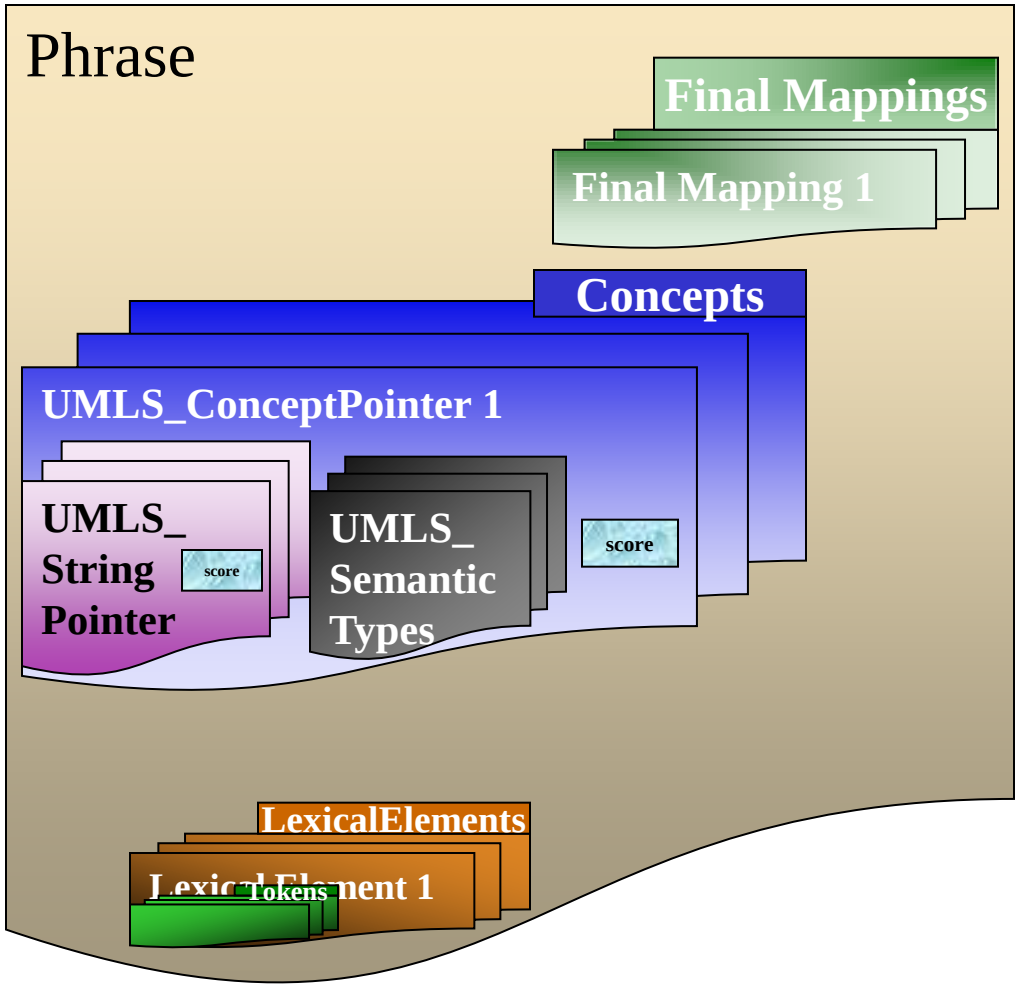
```
// ======================+ Create a Parser object +==
Parser parser = new Parser( argv );
// ============================+ Parse the file +==
Document aDocument = parser.processDocument(aFile);

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() );
}
```



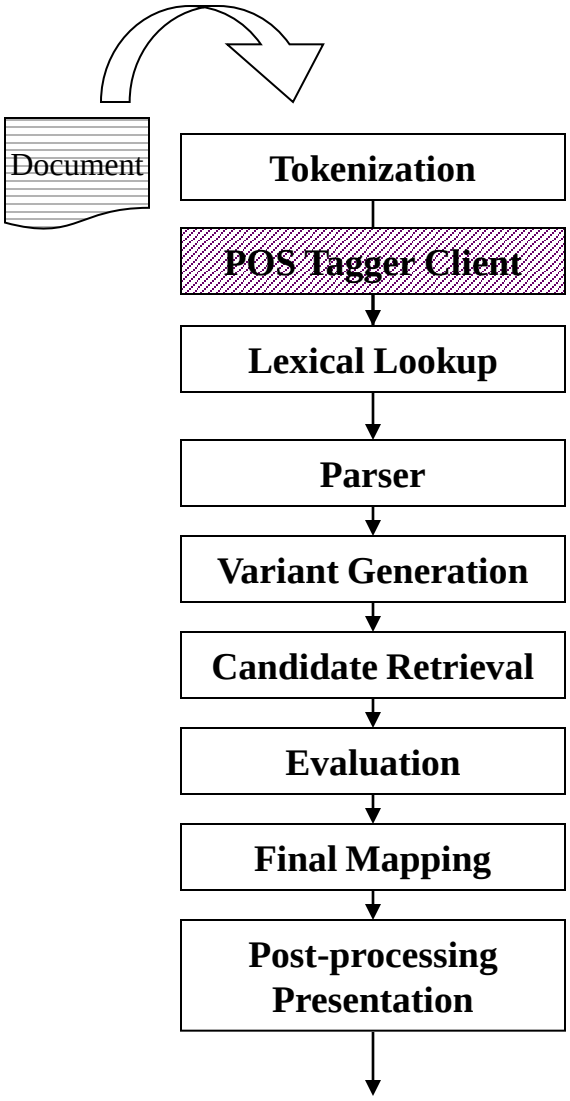
MMTx

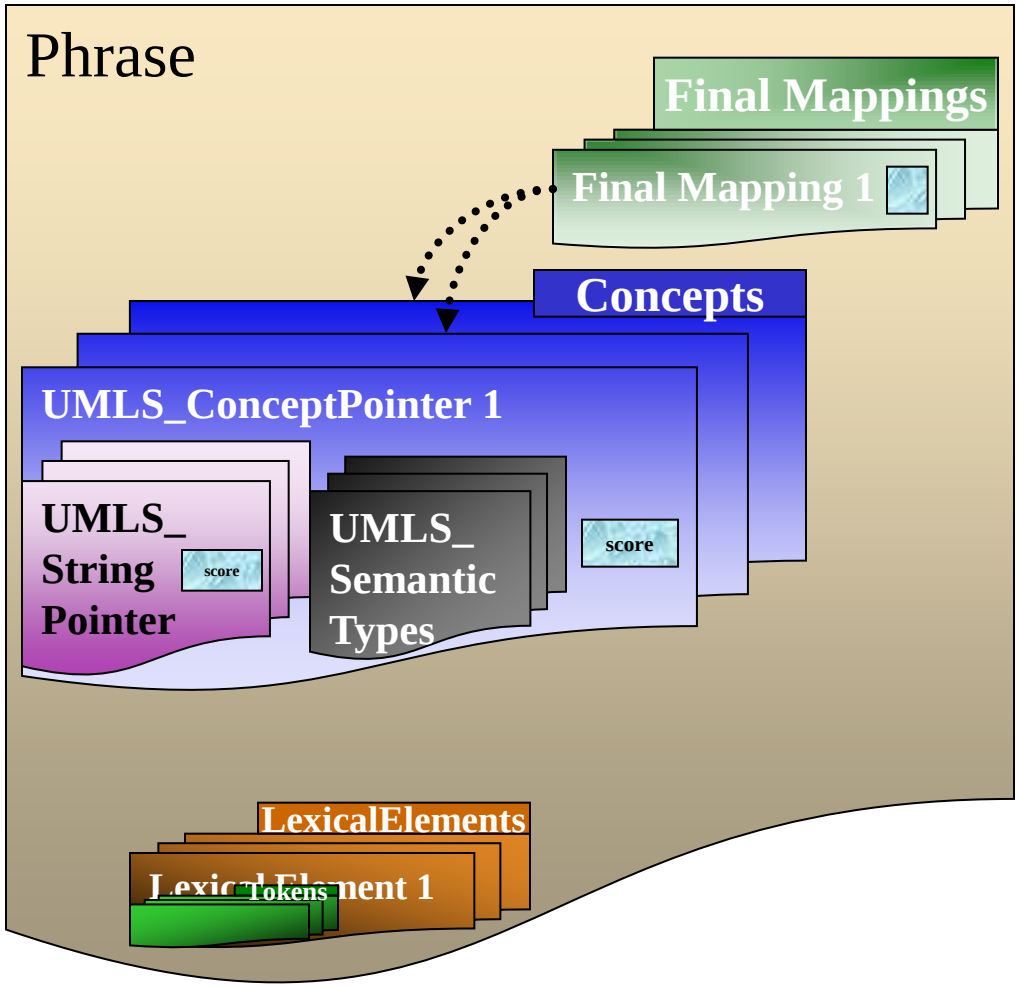
MetaMapTechnology Transfer

- Maps text phrases to Metathesaurus concepts
- Java Implementation of MetaMap



MMTx

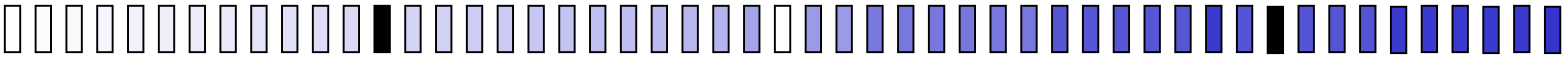

Document
Tokenization
POS Tagger Client
Lexical Lookup
Parser
Variant Generation
Candidate Retrieval
Evaluation
Final Mapping
Post-processing
Presentation


Phrase
Final Mappings
Final Mapping 1
Concepts
UMLS_ConceptPointer 1
UMLS_
String
Pointer
score
UMLS_
Semantic
Types
score
LexicalElements
Tokens

MMTx

Usage

MMTx [<*options*>] [**--fileName**=*infile*]
[**outputFileName**=*outfile*]
--strict_model|**--moderate_model**|**--relaxed_model**
--KSYear=*year*|**--mm_data_version**=*customName*
--threshold=*lowestScore*
--truncate_candidates_mappings
--term_processing|**--allow_overmatches**|**--allow_concept_gaps**
--composite_phrases
--prefer_multiple_concepts
--fielded_output

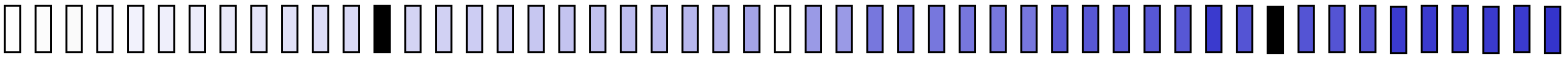


MMTx

```
MMTx --inputFile=5.txt --inputType=freeText
```

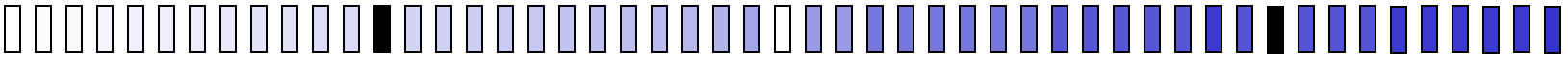
Processing 00000000.tx.3: ***One problem*** *is caused by the VecTest itself, which uses a* ***dipstick*** *to* ***measure*** *the* ***presence*** *of a* ***protein*** *associated with the* ***parasite*** *that causes* ***malaria****.*

Phrase: "**One problem**"
Meta Candidates (2)
  861 Problem, NOS [Finding,Pathologic Function]
  694 One [Quantitative Concept]
Meta Mapping (888)
  694 One [Quantitative Concept]
  861 Problem, NOS [Finding,Pathologic Function]

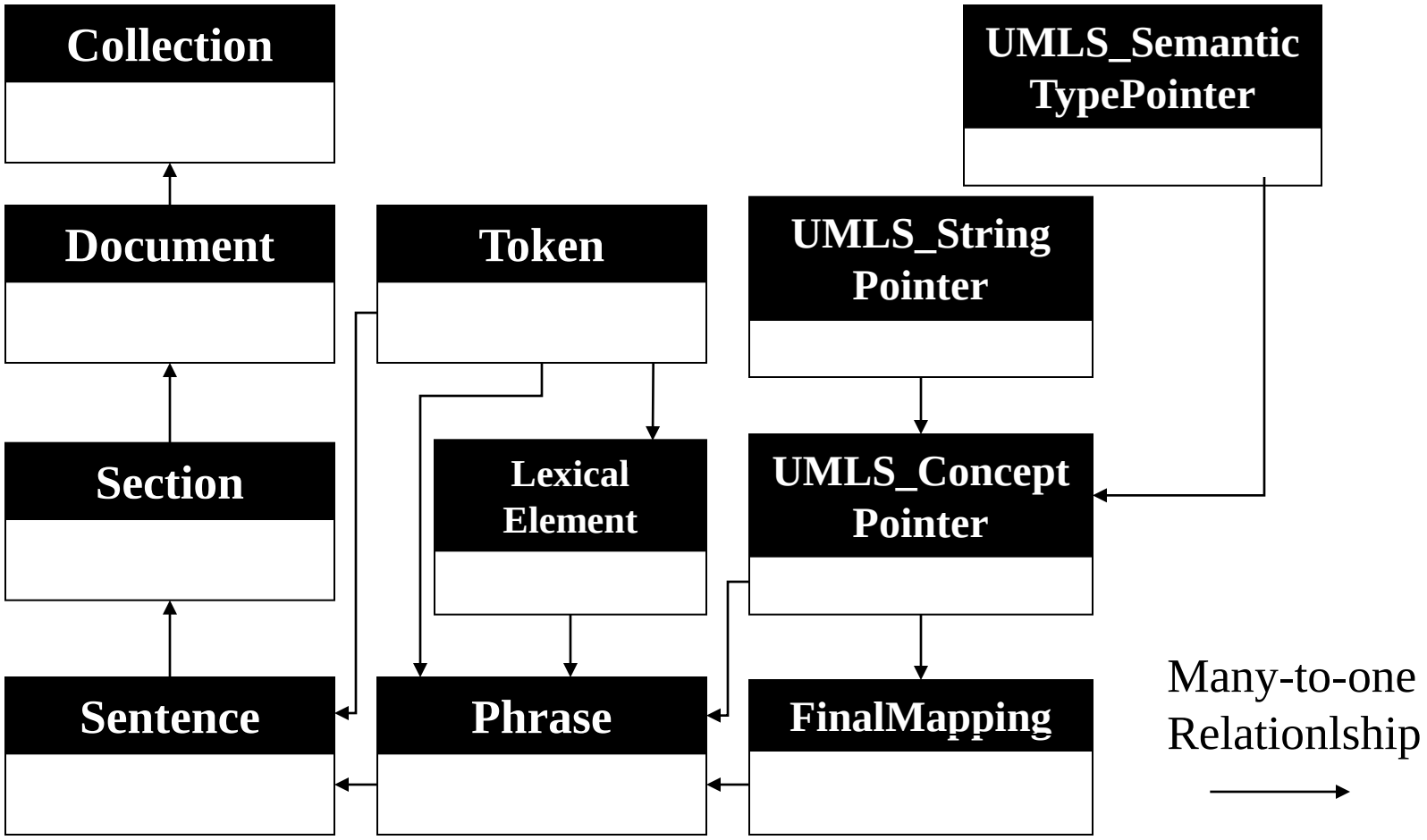


MMTx

```
// ==================+ Create a MMTxAPI object +==
MMTxAPI mmtx = new MMTxAPI( argv );
// ========================+ Analyze the file +==
Document aDocument = mmtx.processDocument(aFile);

Vector      phrases = aDocument.getPhrases() ;
int numberOfPhrases = phrases.size();
Token       aPhrase = null;
// ====================+ Print the Phrases out +==
for ( int i = 0; i < numberOfPhrases; i++ ) {
  aPhrase = (Phrase) phrases.get(i);
  finalConcepts = aPhrase.getFinalMappings();
}
```



Useful Text Feature Classes

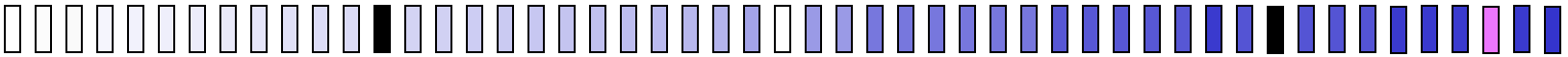


GSpell

"Gottlieb, I think I know why we've been receiving so few commissions."

GSpell

- Spelling suggestion tool
- Pure Java application with Java API’s
- Support for multi word dictionary entries



GSpell: Usage

Usage

GSpellFind.[*sh*|*bat*]

--**dictionary**=*NameOfDictionary*

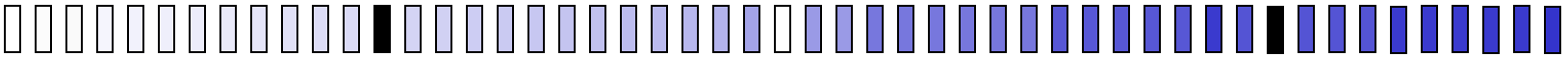
[--**inputFile**=*Source*] [--**outputFile**=*target*]

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

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

[--**fieldedText**] [--**termField**=*X*] [--**correctField**=*Y*]

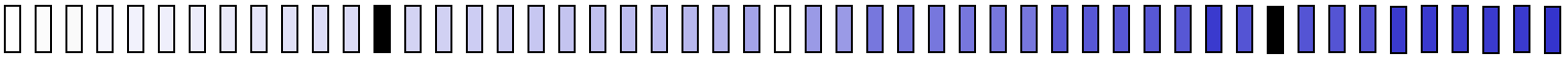
[--reportTime] [--version][--help]



GSpell: Example

Input Term	Suggestion	Edit Distance	Rank	Method	Message

anonomous|**anonymous**|1.0|0.8734230160180236|NGrams|
anonomous|**allonomous**|2.0|0.581967226738810 8|NGrams|
anonomous|**autonomous**|2.0|0.581967226738810 8|NGrams|
anonomous|**anadromous**|3.0|0.295816019208204 8|NGrams|
anonomous|**analogous**|3.0|0.295816019208204 8|NGrams|
anonomous|**anomalous**|3.0|0.295816019208204 8|NGrams|
anonomous|**anonymously**|3.0|0.295816019208248|NGrams|
anonomous|**anonymes**|3.0|0.295816019208204 8|Metaphone|
anonomous|**anonyms**|3.0|0.295816019208204 8|Metaphone|
anonomous|**acoprous**|4.0|0.1147081070210252 1|NGrams|



GSpell: Indexing

Usage

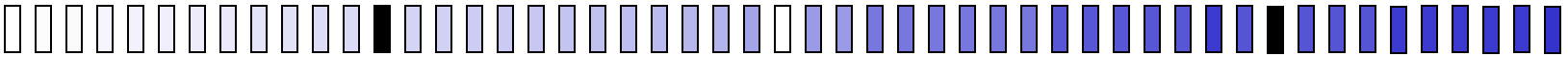
GSpellIndex.[*sh*|*bat*]

--dictionary=*NameOfDictionary*

--inputFile=*SourceFile*

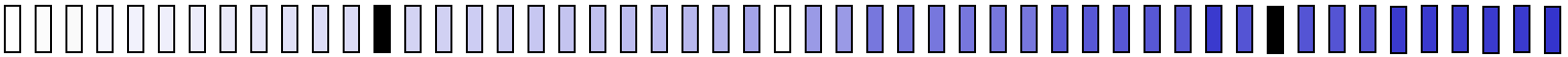
[--reportTime] [--version][--help]

- Format for the input file
  - One word per line



GSpell: Developer's Guide

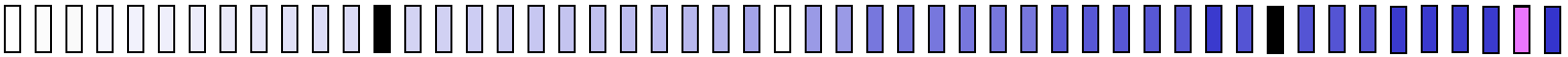
```
import gov.nih.nlm.nls.gspell.GSpell;       // <-------These come from the gspell.jar
import gov.nih.nlm.nls.gspell.Candidate;

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



Downloadable Resources

- umlsks.nlm.nih.gov
- umlsLex.nlm.nih.gov
  - Lvg
  - Java NLP Tools
  - GSpell
- mmtx.nlm.nih.gov *

* Requires a UMLS Licience Aggreement



Lexical Tools for UMLS Developers

November 10, 2002

Allen C. Browne, Guy Divita, Chris Lu

Lister Hill National Center for Biomedical Communications

National Library of Medicine

Lexical Systems: umlsLex.nlm.nih.gov

Email: umlslex@nlm.nih.gov

Knowledge Source Server: http://umlsks.nlm.nih.gov

UMLS Information: http://umlsInfo.nlm.nih.gov

Appendix

NormExample.java

LvgExampleEasy.java

LvgExampleHarder.java

LvgExampleEvenHarder.java

TokenizeExample.java

LexicalLookupExample.java

NpParserExample.java

MMTxExample.java

GSpellExample.java

5.txt

5.tokenized

5.lexicalLookuped

5.parsed

5.mmtxed