NLP Tools
Bulk Data Operation
for Collections - Java 8

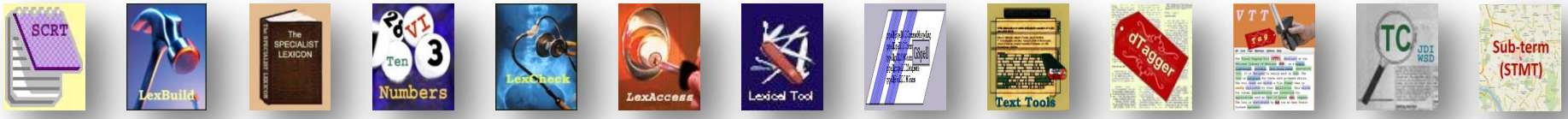
By: Dr. Chris Lu

The Lexical Systems Group

NLM. LHNCBC. CGSB

Dec., 2013

NLP Tools

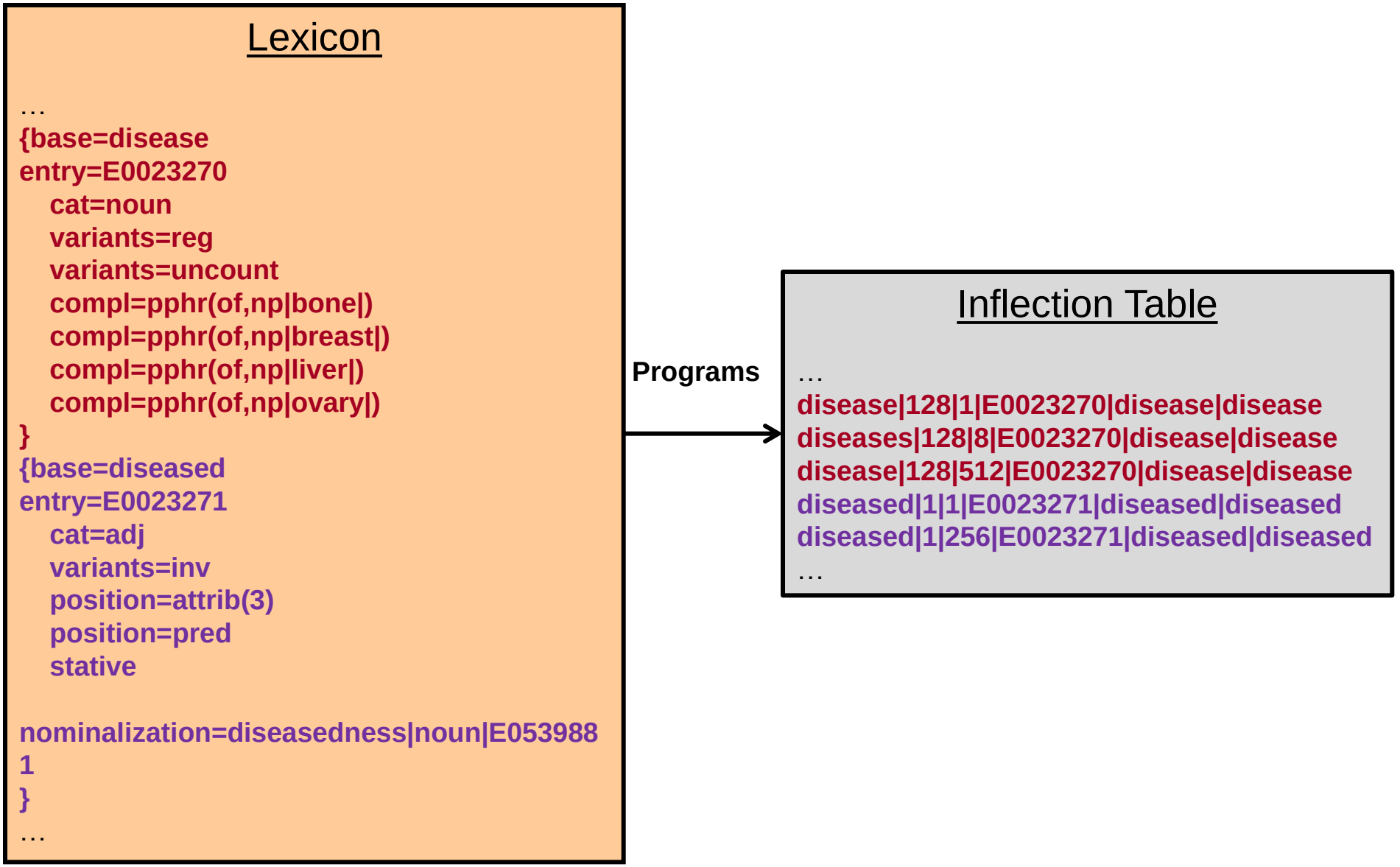


- Lexical Systems Group: http://umlslex.nlm.nih.gov
- The SPECIALIST NLP Tools: http://specialist.nlm.nih.gov

Table of Contents

Bulk Data Operation for Collections (Java 8)

Inflection Table



Inflection Table - Format

- Format (6 fields separated by |):

Field	Name	Notes
1	STR	Inflectional variables
2	SCA	Syntactic category **(value)**
3	INF	Inflection **(value)**
4	EUI	Entry Unique Identifier (EUI) of base
5	BAS	Base form
6	CIT	Citation form

- Category (11 POS):
  - 1: adj, 2: adv, 128: noun, 1024: verb, etc..
- Inflection (24):
  - 1: base,
  - 8: plural, 512: singular (dog|dogs)
  - 256: positive, 2: comparative, 4: superlative (fast|faster|fastest)
  - 1024: infinitive, 128:third person present, 32: past, 64:past participle, 16: present participle, such as “see|sees|saw|seen|seeing”

Inflection table

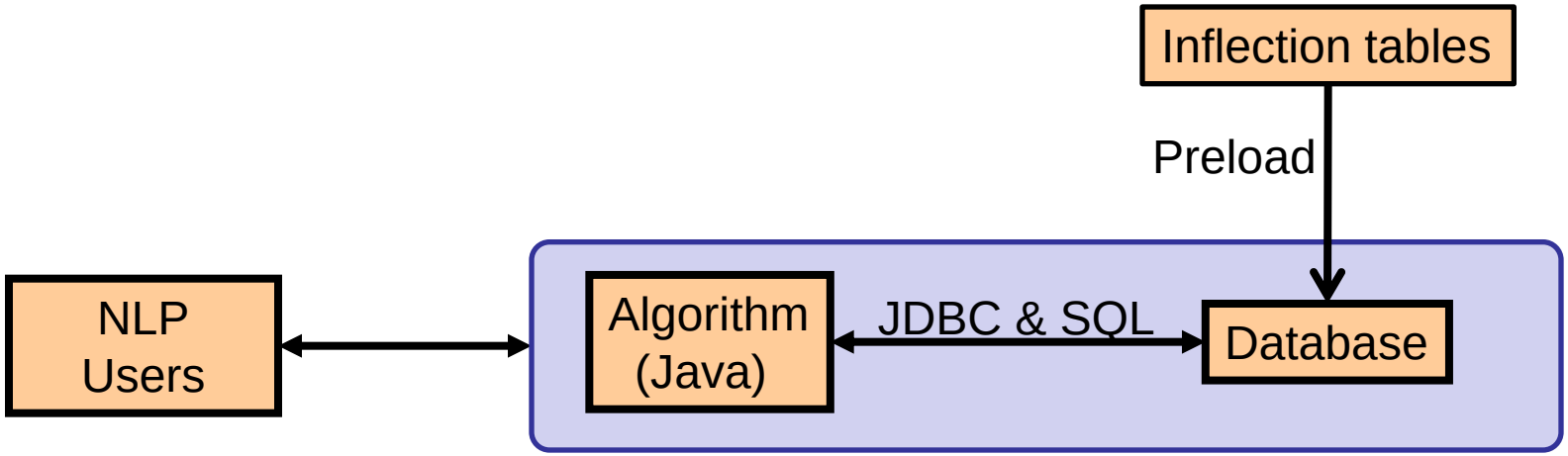
- Example: 1,690,101 records for Lexicon.2014

InflVar	Category	Inflection	EUI	Base	Citation
…	…	…	…	…	…
disease	128	1	E0023270	disease	disease
diseases	128	8	E0023270	disease	disease
disease	128	512	E0023270	disease	disease
diseased	1	1	E0023271	diseased	diseased
diseased	1	256	E0023271	diseased	diseased
…	…	…	…	…	…

LT Flows Use Inflection Table (17/62)

Flow	Descriptions
-f:b	Uninflect a term
-f:B	Uninflect words in a term
-f:Bn	Normalized uninflect words in a term
-f:Ct	Retrieve citation form
-f:d	Generate derivational variants
-f:e	Generate known uninflected from spelling variants
-f:E	Retrieve the unique EUI for a term
-f:f	Filter output to contain only forms from lexicon
-f:G	Generate all fruitful variants
-f:i	Generate inflectional variants
-f:L	Retrieve category and inflection for a term
-f:Ln	Retrieve category and inflection from lexicon
-f:Lp	Retrieve category and inflection for all terms that begin with the given word
-f:N	Normalize the input text in a non-canonical way (Norm)
-f:R	Generate derivational variants, recursively
-f:s	Generate known spelling variants
-f:U	Convert the output of the Xerox Parc stocastic tagger into lvg style pipe delimited format

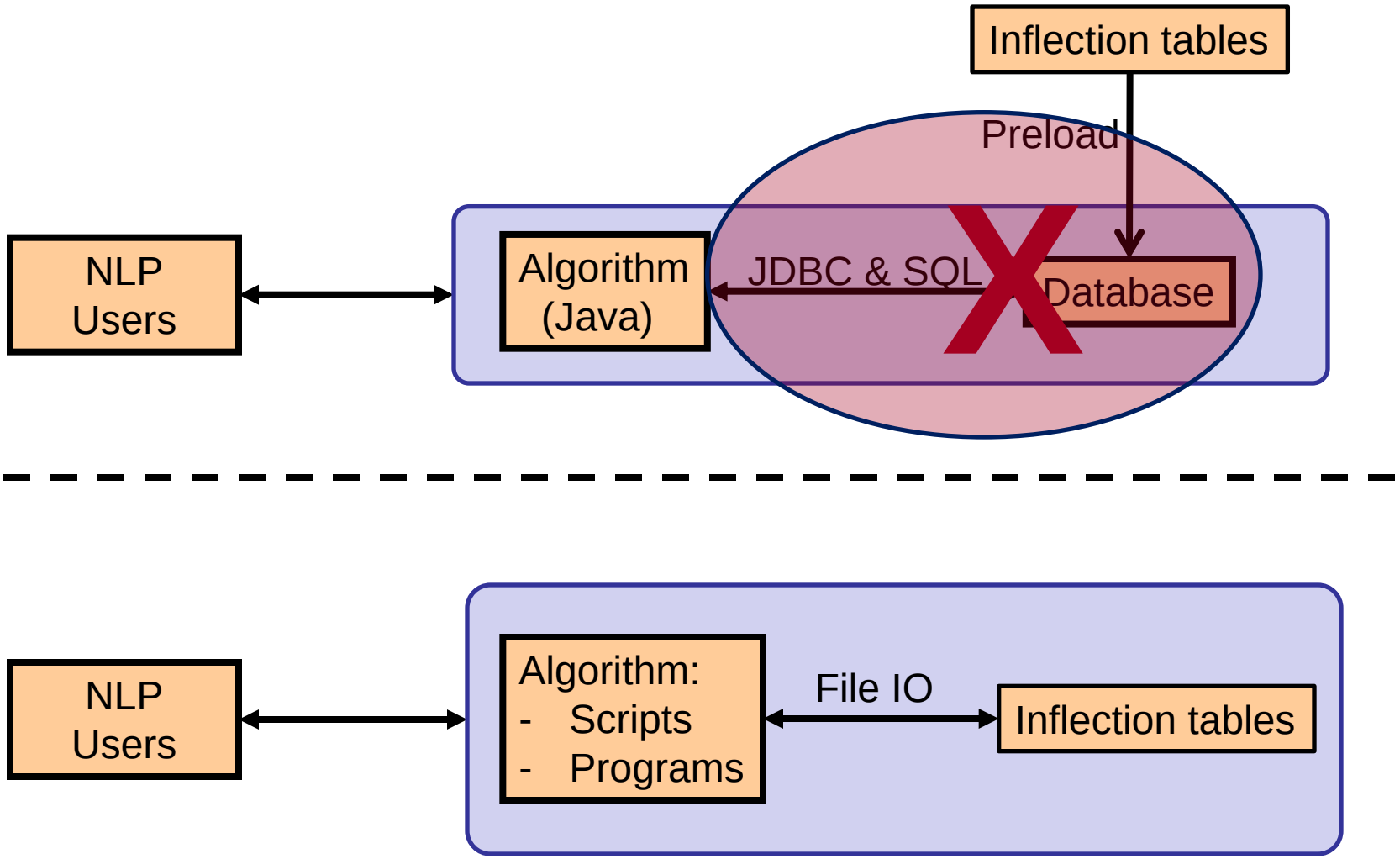
Lexical Tools



Lexical Tools – DB Tasks

- Install a database server – HSqlDb 2.3.0
  - DBA, create DB, design DB tables, etc..
- Utilize JDBC to open/close database connection
  - dbName, dbUrl, dbDriver, etc..
- Use DDL (Data Definition Language) – create DB tables, shutdown, etc..
  - `CREATE TABLE Inflection (inTerm varchar(105) NOT NULL, termCat INT, termInfl INT, eui varchar(8) NOT NULL, unTerm varchar(105) NOT NULL, ctTerm varchar(105));`
  - `CREATE INDEX index1 on Inflection (eui);`
  - `DROP TABLE Inflection;`
- Use SQL (Structured Query Language) for query by prepared statement:
  - `INSERT INTO Inflection (ifTerm, termCat, termInfl, eui, unTerm,ctTerm) Values (?, ?, ?, ?, ?, ?)`
  - `SELECT * from Infelction where ifTerm = 'disease'`
- Other issues: DB APIs, DB Objects, exception handling, etc..

A Simpler Way for Simple Task?



Ex-1: Iteration - Reformat

- Task: convert categories and inflections to readable names

InflVar	Category	Inflection	EUI	Base	Citation
…	…	…	…	…	…
disease	128 -> noun	1 -> base	E0023270	disease	disease
diseases	128 -> noun	8 -> plur	E0023270	disease	disease
disease	128 -> noun	512 -> sing	E0023270	disease	disease
diseased	1 -> adj	1 -> base	E0023271	diseased	diseased
diseased	1 -> adj	256 -> positive	E0023271	diseased	diseased
…	…	…	…	…	…

Ex-1: Iteration

- Change to readable format (replace value by name for categories and inflections)
  1) Go through all records
  2) For each record, change to readable format (use name for value)
  3) Print out readable format

- Notes:
  - All records are read in from a file (inflVar.data) and stored in a collection of InflVar objects (List <InflVar> records)
  - InflVar is a Java class of inflectional variable object with basic methods
  - All source codes will be available on Lexical Tools Web Site

Ex-1: Iteration (Java 7 - for)

- Change to readable format (replace value by name for categories and inflections)
  1) Go through all records
  2) For each record, change to readable format (use name for value)
  3) Print out readable format

Java 7 (external iterator):

```
for(InflVar rec:records)
{
    String recStr = rec.ToReadableFormat();
    PrintOut(recStr);
}
```

Ex-1: Java 8 - ForEach (T -> void)

- Task: Change to readable format (replace value by name for categories and inflections)
  1) Go through all records (stream)
  2) For each records, print out readable format (Lambda Expression)

Java 8 (internal iterator):

```
records.stream()
   .forEach(rec -> PrintOut(rec.ToReadableFormat()));
```

Ex-1: ForEach (Consumer<T>)

```
records.stream()
   .map(rec -> rec.ToReadableFormat())
   .forEach(recStr -> PrintOut(recStr));
```

```
records.stream()
   .forEach(rec -> PrintOut(rec.ToReadableFormat()));
```

```
records.stream()
   .forEach(PrintReadableFormat());

public static Consumer<InflVar> PrintReadableFormat()
{
    return rec -> PrintOut(rec.ToReadableFormat());
}
```

Ex-1: Iteration Result

- Task: retrieve all records and convert to readable names for categories and inflections (1,690,181)

InflVar	Category	Inflection	EUI	Base	Citation
…	…	…	…	…	…
disease	noun	base	E0023270	disease	disease
diseases	noun	plur	E0023270	disease	disease
disease	noun	sing	E0023270	disease	disease
diseased	adj	base	E0023271	diseased	diseased
diseased	adj	positive	E0023271	diseased	diseased
…	…	…	…	…	…

Ex-2: Filter

- Task: retrieve all records that are ***base forms and category are adverb***
  - Field 2: category is “adv”
  - Field 3: inflection is “base”

Ex-2: Filter (Java 7 – if)

- Task: retrieve all records that are ***base forms and adv***
  1) Go through all records
  2) Filter out records are not adv
  3) Filter out records are not base forms
  4) Print out readable format

Java 7 (if):

```
for(InflVar rec:records)
{
    if((InflVar.IsAdv(rec) == true)
    && (InflVar.IsBase(rec) == true))
    {
        PrintOut(rec.ToReadableFormat());
    }
}
```

Ex-2: Java 8 - Filter (T -> boolean)

- Task: retrieve all records that are ***base forms and adv***
  1) Go through all records
  2) Filter out not adv (Lambda expression)
  3) Filter out not base forms (method references)
  4) Print out readable format

Java 8 (filter):

```
records.stream()
   .filter(rec -> InflVar.IsAdv(rec))
   .filter(InflVar::IsBase)
   .forEach(rec -> PrintOut(rec.ToReadableFormat()));
```

Ex-2: Filter Results

- Task: retrieve all records that are ***base forms and categories are adverb (13,222)***

InflVar	Category	Inflection	EUI	Base	Citation
…	…	…	…	…	…
aboard	adv	base	E0006522	aboard	aboard
aborad	adv	base	E0006532	aborad	aborad
about	adv	base	E0006547	about	about
above	adv	base	E0006550	above	above
abroad	adv	base	E0006570	abroad	abroad
…	…	…	…	…	…

Ex-2a: Filters

- Task: retrieve all records that are ***base forms and categories are either adverb or adjective***
  1) Go through all records
  2) Filter out records that inflections are not base forms
  3) Filter out records that categories are not adv or adj
  4) Print out readable format

Ex-2a: Filters (Java 7)

- Task: retrieve all records that are ***base forms and categories are either adverb or adjective***
  1) Go through all records
  2) Filter out records that inflections are not base forms
  3) Filter out records that categories are not adv or adj
  4) Print out readable format

Java 7 (if):

```
for(InflVar rec:records)
{
    if((InflVar.IsBase(rec) == true)
    && ((InflVar.IsAdv(rec) == true)
     || (InflVar.IsAdj(rec) == true)))
    {
        PrintOut(rec.ToReadableFormat());
    }
}
```

Ex-2a: Java 8 - Filters

- Task: retrieve all records that are ***base forms and categories are either adverb or adjective***

Java 8 (filters):

```
records.stream()
   .filter(rec -> InflVar.IsBase(rec))
   .filter(rec -> InflVar.IsAdv(rec)
    || InflVar.IsAdj(rec))
   .forEach(rec -> PrintOut(rec.ToReadableFormat()));
```

```
records.stream()
   .filter(rec -> InflVar.IsBase(rec)
    && (InflVar.IsAdv(rec) || InflVar.IsAdj(rec)))
   .forEach(rec -> PrintOut(rec.ToReadableFormat()));
```

Ex-2a: Java 8 - Filter (Predicate<T>)

```
records.stream()
   .filter(rec -> InflVar.IsBase(rec)
      && (InflVar.IsAdv(rec) || InflVar.IsAdj(rec)))
   .forEach(rec -> PrintOut(rec.ToReadableFormat()));
```

```
records.stream()
   .filter(MatchAdjAdvBase())
   .forEach(PrintReadableFormat());

Private static Predicate<InflVar> MatchAdjAdvBase()
{
    return filter(rec -> InflVar.IsBase(rec)
       && (InflVar.IsAdv(rec) || InflVar.IsAdj(rec));
}
```

Ex-2a: Filters Results

- Task: retrieve all records that are ***base forms and categories are either adverb or adjective*** ***(111,675)***

InflVar	Category	Inflection	EUI	Base	Citation
…	…	…	…	…	…
aboard	adv	base	E0006522	aboard	aboard
abominable	adj	base	E0006529	abominable	abominable
aborad	adv	base	E0006532	aborad	aborad
aborad	adj	base	E0006533	aborad	aborad
aboral	adj	base	E0006534	aboral	aboral
…	…	…	…	…	…

Ex-3: Map

- Task: retrieve all ***base forms*** that are either adv or adj
  1) Go through all records
  2) Filter out not base forms
  3) Filter out not adv or adj
  4) Get the ***base form*** (map from InflVar record to base from)
  5) Print out base forms

Ex-3: Map (Java 7 – getter)

- Task: retrieve all base forms that are either adv or adj

Java 7 (getter):

```
for(InflVar rec:records)
{
    if((InflVar.IsBase(rec) == true)
    && ((InflVar.IsAdj(rec) == true)
     || (InflVar.IsAdv(rec) == true)))
    {
        String base = rec.GetBase();
        PrintOut(base);
    }
}
```

Ex-3: Java 8 - Map (T -> R)

- Task: retrieve all base forms that are either adv or adj

Java 8 (map):

```
records.stream()
   .filter(MatchAdjAdvBase())
   .map(rec -> rec.GetBase())
   .forEach(base -> PrintOut(base));
```

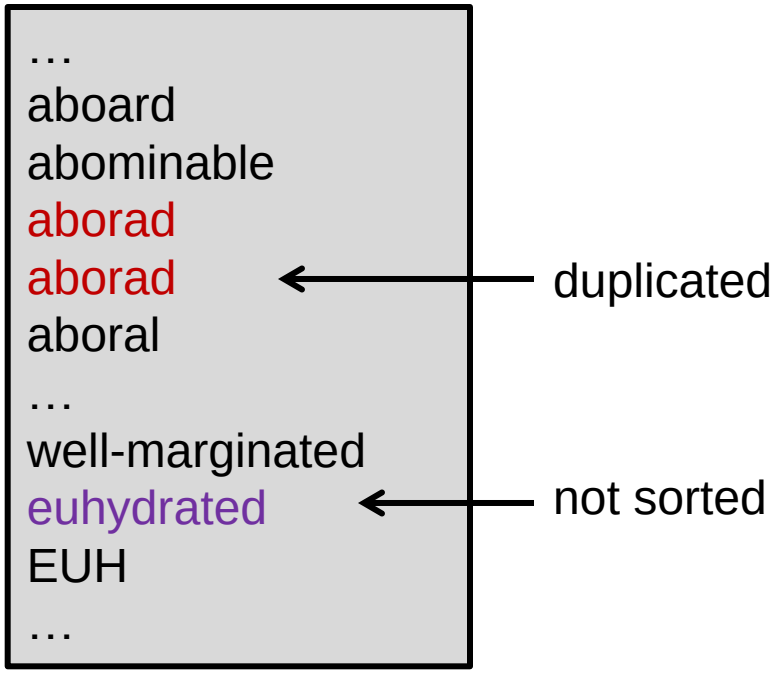
Ex-3: Java 8 - Map (Function<T, R>)

```
records.stream()
   .filter(MatchAdjAdvBase())
   .map(rec -> rec.GetBase())
   .forEach(base -> PrintOut(base));
```

```
records.stream()
   .filter(MatchAdjAdvBase())
   .map(MapToBase())
   .forEach(base -> PrintOut(base));

public static Function<InflVar, String> MapToBase()
{
    return rec -> rec.GetBase();
}
```

Ex-3: Map - Result

- Task: retrieve all base forms that are either adv or adj



Ex-3a: Distinct & Sort

- Task: Retrieve all base forms that are either adv or adj, distinct and sort
  1) Go through all records
  2) Filter out not base forms
  3) Filter out not adv or adj
  4) Get the base form (map from record to base from)
  5) ***Distinct & sorted***
  6) Print out base form

Ex-3a: Distinct & Sort

- Task: Retrieve all base forms that are either adv or adj, distinct and sort

Java 7 (TreeSet):

```
TreeSet<String> uSortList = new TreeSet<String>();
for(InflVar rec:records)
{
    if((InflVar.IsBase(rec) == true)
    && ((InflVar.IsAdj(rec) == true)
     || (InflVar.IsAdv(rec) == true)))
    {
        String base = rec.GetBase();
        uSortList.add(base);
    }
    for(String base:uSortList)
    {
       PrintOut(base);
    }
}
```

Ex-3a: Java 8 - Distinct & Sort

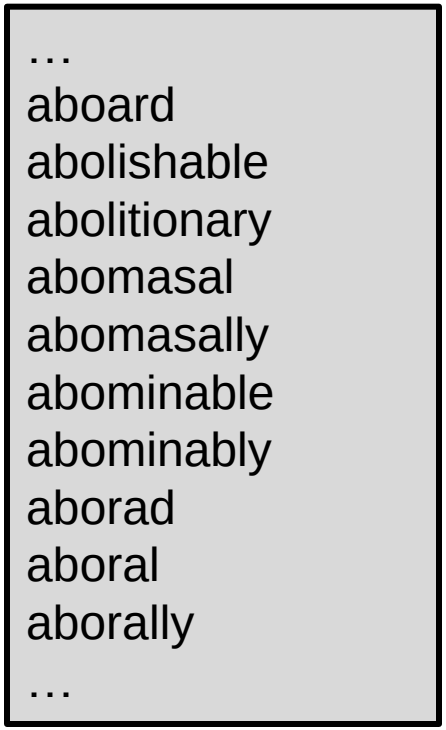
- Task: Retrieve all base forms that are either adv or adj, distinct and sort

Java 8: distinct & sort

```
records.stream()
   .filter(MatchAdjAdvBase())
   .map(rec -> rec.GetBase())
   .distinct()
   .sorted()
   .forEach(base -> PrintOut(base));
```

Ex-3a: Distinct & Sort Result

- Task: Retrieve all base forms that are either adv or adj, distinct and sort



Ex-3b: Sort (Comparator<T>)

- Find all records that are base forms and either adv or adj
  1) Go through all records
  2) Filter out not base forms
  3) Filter out not adv or adj
  4) Distinct
  5) ***Sorted by category***
  6) Print out readable format

Java 8: sort (Comparator)

```
records.stream()
   .filter(MatchAdjAdvBase())
   .distinct()
   .sorted(Comparator.comparing(InflVar::getCat))
   .forEach(rec -> PrintOut(rec.ToReadableFormat()));
```

Ex-3b: Sort by Cat - Result

```
…
abominable|adj|base|E0006529|abominable|abominable
aborad|adj|base|E0006533|aborad|aborad
aboral|adj|base|E0006534|aboral|aboral
…
abomasal|adj|base|E0420408|abomasal|abomasal
…
aboard|adv|base|E0006522|aboard|aboard
aborad|adv|base|E0006532|aborad|aborad
about|adv|base|E0006547|about|about
..
abomasally|adv|base|E02344588|abomasally|abomasally
…
```

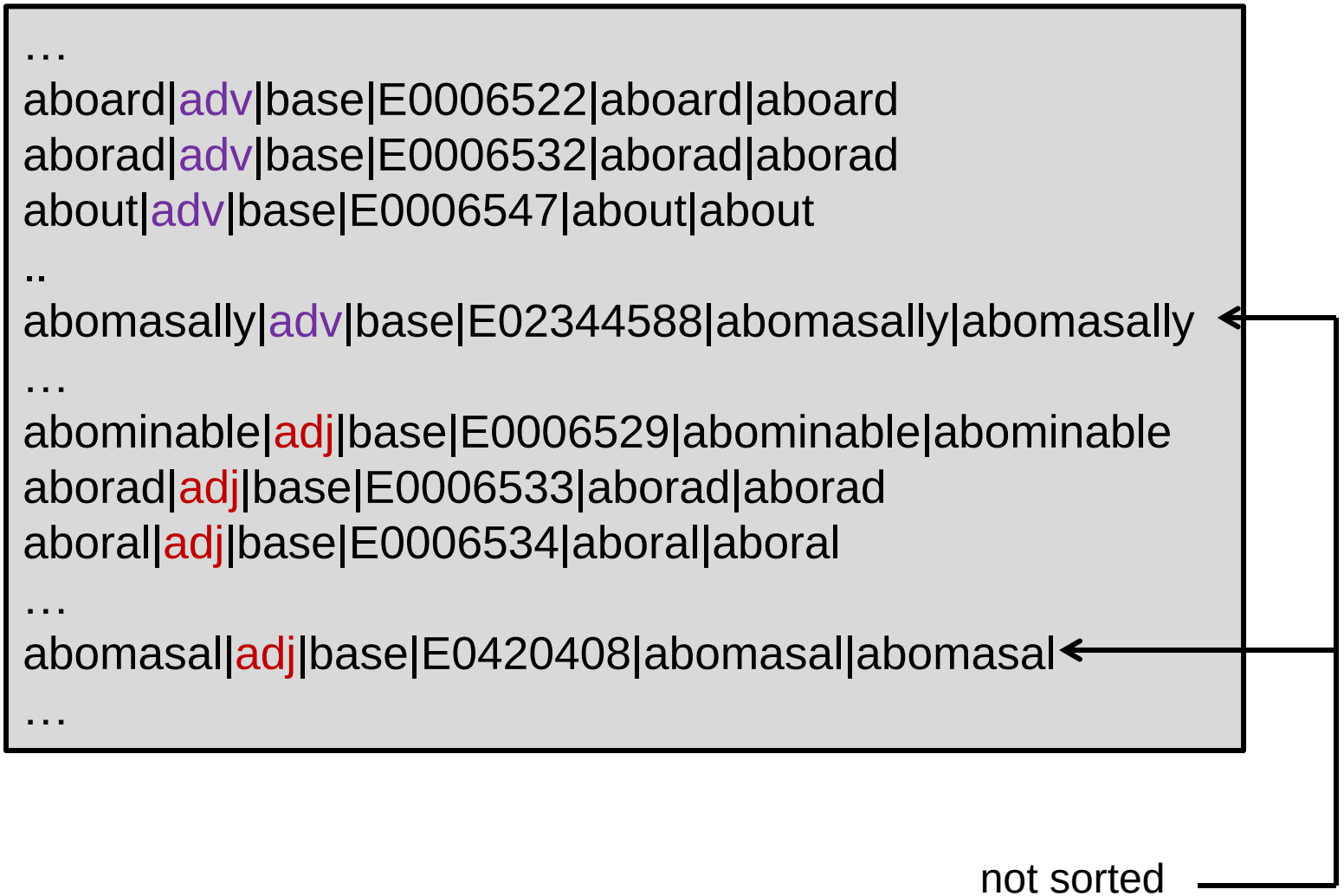
Ex-3c: Reversed Sort

- Find all base forms that are either adv or adj
  1) Go through all records
  2) Filter out not base forms
  3) Filter out not adv or adj
  4) Distinct
  5) Sorted by category, reversed
  6) Print out readable format

Java 8: reversed sort

```
records.stream()
  .filter(MatchAdjAdvBase())
  .distinct()
  .sorted(
     Comparator.comparing(InflVar::getCat).reversed())
  .forEach(rec -> PrintOut(rec.ToReadableFormat()));
```

Ex-3c: Reversed Sort - Result



Ex-3d: Composed Comparator

- Find all base forms that are either adv or adj
  1) Go through all records
  2) Filter out not base forms
  3) Filter out not adv or adj
  4) Distinct
  5) ***Sorted by category, then sorted by inflVar***
  6) Print out

Java 8: sort (Composed Comparator)

```
records.stream()
  .filter(MatchAdjAdvBase())
  .distinct()
  .sorted(Comparator.comparing(InflVar::getCat)
    .thenComparing(Comparator.comparing(InflVar::getInflVar))
  .forEach(rec -> PrintOut(rec.ToReadableFormat()));
```

Ex-3d: Composed Sort - Result

```
…
abomasal|adj|base|E0420408|abomasal|abomasal
abominable|adj|base|E0006529|abominable|abominable
aborad|adj|base|E0006533|aborad|aborad
aboral|adj|base|E0006534|aboral|aboral
…
aboard|adv|base|E0006522|aboard|aboard
abomasally|adv|base|E02344588|abomasally|abomasally
abominably|adv|base|E0551676|abominably|abominably
aborad|adv|base|E0006532|aborad|aborad
aborally|adv|base|E0421958|aborally|aborally
…
```

Ex-4: Collector

- Task: find all categories of a term
  1) Go through all records
  2) Find matched inflection variant
  3) Add the category name (distinct & sort)
  4) Print out results with a , between

- Example:
  - saw => noun, verb
  - left => adj, adv, noun, verb

Ex-4: Collector (Java 7)

```
TreeSet<String> catSet = new TreeSet<String>();
for(InflVar rec:records)
{
    if(rec.GetInflVar().equals(inTerm))
    {
        String cat = rec.GetCatName();
        catList.add(cat);
    }
}
String outStr = new String();
Iterator<String> it = catSet.iterator();
int curIndex = 0;
Whil((it.hasNext()) && (curIndex < carSet.size()-1))
{
   outStr += it.next() + ", ";
   curIndex++;
}
if(catList.size() > 0)
{
  outStr += it.next();
}
```

Ex-4: Java 8 - Collect

- Task: find all categories of a term

Java 8 (Collect):

```
String outStr = records.stream()
   .filter(rec -> rec.GetInflVar().equals(inTerm))
   .map(rec -> rec.GetCatName())
   .distinct()
   .sorted()
   .collect(joining(", "));
```

- Example:
  - saw => noun, verb
  - left => adj, adv, noun, verb

Ex-4: Java 8 - Collect (Parallelism)

- Task: find all categories of a term
- Parallelism: multiple CPUs for better performance
  - Java 7: fork-join
  - Java 8: parallel or parallelStream

Java 8 (Collect):

```
String outStr = records.parallelStream()
  .filter(rec -> rec.GetInflVar().equals(inTerm))
  .map(rec -> rec.GetCatName())
  .distinct()
  .sorted()
  .collect(joining(", "));
```

Ex-5: Map & Collect

- Task: find terms with most categories in Lexicon
  1) Map records to key-value pairs with key as inflVar and values as sorted distinct categories

Key (inflVar)	Values (Categories)
…	…
left	adj, adv, noun, verb
…	…
saw	noun, verb
…	…

  2) Find key-value pairs with most categories
  3) Print results
     - Format: inflVar: [cat1, cat2, …]
     - Sorted by inflVar

Ex-5: Map & Collect – Java 7

```
TreeMap<String, TreeSet<String>> termCats = new TreeMap<String,
TreeSet<String>>();
for(InflVar rec:records)
{
    String inflVar = rec.GetInflVar();
     String catName = rec.GetCatName();
      if(termCats.containsKey(inflVar) == false)  // new
      {
          TreeSet<String> catList = new TreeSet<String>();
          catList.add(catName);
          termCats.put(inflVar, catList);
      }
      else    // exist
      {
          termCats.get(inflVar).add(catName);
      }
}
Set<String> terms = termCats.keySet();
int maxCatNo = 0;
ArrayList<String> resultList = new ArrayList<String>();
for(String term:terms)
{
    TreeSet<String> sortedCatList = termCats.get(term);
     if(sortedCatList.size() > maxCatNo) // find a new term with most
     {
          maxCatNo = sortedCatList.size();
          resultList = new ArrayList<String>();
          String resultStr = term + ": " + sortedCatList.toString();
          resultList.add(resultStr);
      }
      else if(sortedCatList.size() == maxCatNo)
      {
          String resultStr = term + ": " + sortedCatList.toString();
          resultList.add(resultStr);
      }
}
String outStr = new String();
for(String resultStr:resultList)
{
    outStr += resultStr + "\n";
}
```

Ex-5: Java 8 - Map & Collect

```
long maxCatNo = records.parallelStream()
    .collect(groupingBy(InflVar::getInflVar, mapping(InflVar::getCatName, toSet())))
    .entrySet()
    .parallelStream()
    .mapToLong(Ex6_CollectToMap::GetCatNum) // find the cat number
    .max()
    .getAsLong();

String outStr = records.parallelStream()
    .collect(groupingBy(InflVar::getInflVar, TreeMap::new,
mapping(InflVar::getCatName, toSet())))
    .entrySet()
    .parallelStream()
    .filter(me -> Ex6_CollectToMap.CatNumFilter(me, maxCatNo))
    .map(Ex6_CollectToMap::GetKeyValueStr)
    .collect(joining("\n"));
```

Ex-5: Java 8 - Map & Collect

```
long maxCatNo = records.parallelStream()
    .collect(groupingBy(InflVar::getInflVar,
                        mapping(InflVar::getCatName, toSet())))
    .entrySet()
    .parallelStream()
    .mapToLong(Ex6_CollectToMap::GetCatNum) // find the cat number
    .max()
    .getAsLong();

String outStr = records.parallelStream()
    .collect(groupingBy(InflVar::getInflVar, TreeMap::new,
                        mapping(InflVar::getCatName, toSet())))
    .entrySet()
    .parallelStream()
    .filter(me -> Ex6_CollectToMap.CatNumFilter(me, maxCatNo))
    .map(Ex6_CollectToMap::GetKeyValueStr)
    .collect(joining("\n"));
```

Ex-5: Map & Collect – Result

- Task: find terms with most categories in Lexicon (3)

last: [adj, adv, det, noun, verb]
round: [adj, adv, noun, prep, verb]
up: [adj, adv, noun, prep, verb]

Ex-6: Reduce

- Task: find all terms has 4 categories of adj, adv, noun, verb
  1) Map records to key-value pairs with key as inflVar and values as distinct categories
  2) Filter out those does not has specified 4 categories
  3) Print out results (inflVar)
     - Distinct & sort

Ex-6: Reduce – Java 7

```
Hashtable<String, ArrayList<InflVar>> recMap = new Hashtable<String,
ArrayList<InflVar>>();
for(InflVar rec:records)
{
    String inflVar = rec.GetInflVar();
    if(recMap.containsKey(inflVar) == false) // not exist - new
    {
        ArrayList<InflVar> inflVarList = new ArrayList<InflVar>();
        inflVarList.add(rec);
        recMap.put(inflVar, inflVarList);
    }
    else
   {
        recMap.get(inflVar).add(rec);
    }
}
ArrayList<String> sortList = new ArrayList<String>();
Enumeration<String> keyE = recMap.keys();
while(keyE.hasMoreElements())
{
    String key = keyE.nextElement();
    long value = 0;
    ArrayList<InflVar> inflVarList = recMap.get(key);
    for(InflVar inflVarItem:inflVarList)
    {
        value = value | getCatBitValue(inflVarItem);
    }
    if(value == 15)
    {
        sortList.add(key);
    }
}
Collections.sort(sortList);
for(String term:sortList)
{
    Ex0_FileIo.PrintToFile(term, writer);
}
```

Ex-6: Java 8 - Reduce

```
records.parallelStream()
    .collect(groupingBy(InflVar::getInflVar))
    .entrySet()
    .stream()
    .filter(Ex7_Reduce::IsAanv)
    .map(Map.Entry::getKey)
    .sorted()
    .forEach(Ex0_FileIo.PrintStrToFile(writer));


private static boolean IsAanv(Map.Entry<String, List<InflVar>> entry)
{
    long bitValue = (entry.getValue()
        .parallelStream()
        .mapToLong(rec -> Ex7_Reduce.getCatBitValue(rec))
        .reduce(0, (l1, l2) -> l1 | l2));
    return (bitValue == 15);
}
```

Ex-6: Java 8 - Reduce

```
records.parallelStream()
    .collect(groupingBy(InflVar::getInflVar))
    .entrySet()
    .stream()
    .filter(Ex7_Reduce::IsAanv)
    .map(Map.Entry::getKey)
    .sorted()
    .forEach(Ex0_FileIo.PrintStrToFile(writer));

private static boolean IsAanv(Map.Entry<String,
    List<InflVar>> entry)
{
    long bitValue = (entry.getValue()
        .parallelStream()
        .mapToLong(rec -> Ex7_Reduce.getCatBitValue(rec))
        .reduce(0, (l1, l2) -> l1 | l2));
    return (bitValue == 15);
}
```

Ex-6: Reduce – Result

- Task: find all terms has 4 categories of adj, adv, noun, verb (64)

…
firm
flush
forward
free
full
home
last
left
level
light
long
low
…

Flows Use Inflection tables (17/62)

Flow	Descriptions
-f:b	Uninflect a term
-f:B	Uninflect words in a term
-f:Bn	Normalized uninflect words in a term
-f:Ct	Retrieve citation form
-f:d	Generate derivational variants
-f:e	Generate known uninflected from spelling variants
-f:E	Retrieve the unique EUI for a term
-f:f	Filter output to contain only forms from lexicon
-f:G	Generate all fruitful variants
-f:i	Generate inflectional variants
-f:L	Retrieve category and inflection for a term
-f:Ln	Retrieve category and inflection from lexicon
-f:Lp	Retrieve category and inflection for all terms that begin with the given word
-f:N	Normalize the input text in a non-canonical way (Norm)
-f:R	Generate derivational variants, recursively
-f:s	Generate known spelling variants
-f:U	Convert the output of the Xerox Parc stocastic tagger into lvg style pipe delimited format

Ex-7: Contain

- Task: Find if a term is in Lexicon
  1) Go through all records
  2) Find matched inflectional variant
  3) Return results (boolean)

- Example
  - diabetics => true
  - Laurent => false

Ex-7: Contain (Java 7 – if)

- Task: find if a term is in Lexicon

Java 7 (if):

```
boolean flag = false;
for(InflVar rec:records)
{
    if(rec.GetInflVar().equals(inTerm))
    {
        flag = true;
        break;
    }
}
```

Ex-7: Java 8 - Contain

- Task: find if a term is in Lexicon

Java 8 (contain):

```
boolean flag = records.stream()
   .filter(rec -> rec.GetInflVar().equals(inTerm))
   .findFirst()
   .isPresent();
```

Ex-7a: Parallelism

- Task: find if a term is in Lexicon
- Parallelism: multiple CPUs for better performance
  - Java 7: fork-join
  - Java 8: parallel or parallelStream

Java 8 (parallelism):

```
boolean flag = records.parallelStream()
   .filter(rec -> rec.GetInflVar().equals(inTerm))
   .findFirst()
   .isPresent();
```

Flows Use Inflection tables (17/62)

Flow	Descriptions
-f:b	Uninflect a term
-f:B	Uninflect words in a term
-f:Bn	Normalized uninflect words in a term
-f:Ct	Retrieve citation form
-f:d	Generate derivational variants
-f:e	Generate known uninflected from spelling variants
-f:E	Retrieve the unique EUI for a term
-f:f	Filter output to contain only forms from lexicon
-f:G	Generate all fruitful variants
-f:i	Generate inflectional variants
-f:L	Retrieve category and inflection for a term
-f:Ln	Retrieve category and inflection from lexicon
-f:Lp	Retrieve category and inflection for all terms that begin with the given word
-f:N	Normalize the input text in a non-canonical way (Norm)
-f:R	Generate derivational variants, recursively
-f:s	Generate known spelling variants
-f:U	Convert the output of the Xerox Parc stocastic tagger into lvg style pipe delimited format

Ex-8: Java 8 - Base

- Task: find base from of a term
  1) Find matched records (with matching inflVar)
  2) Map inflVar to base from
  3) Print base form (distinct & sort)

- Example:
  - Input: saw
  - Output: saw, see

Java 8:

```
String outStr = records.parallelStream()
  .filter(rec -> rec.GetInflVar().equals(inTerm))
  .map(rec -> rec.GetBase())
  .distinct()
  .sort()
  .collect(joining(", "));
```

Ex-8a: Java 8 - Citation

- Task: find the citation form of a term
  1) Find matched records (with matching inflVar)
  2) Map inflVar to citation from
  3) Print citation form (distinct & sort)

- Example:
  - Input: colour
  - Output: color

Java 8:

```
String outStr = records.parallelStream()
  .filter(rec -> rec.GetInflVar().equals(inTerm))
  .map(rec -> rec.GetCitation())
  .distinct()
  .sort()
  .collect(joining(", "));
```

Ex-8b: Java 8 - EUI

- Task: find the Entry Unique Identifier (EUI) form of a term
  1) Find matched records (with matching inflVar)
  2) Map inflVar to EUI
  3) Print EUI (distinct & sort)

- Example:
  - Input: saw
  - Output: E0054443, E0054444, E0055007

Java 8:

```
String outStr = records.parallelStream()
  .filter(rec -> rec.GetInflVar().equals(inTerm))
  .map(rec -> rec.GetEui())
  .distinct()
  .sort()
  .collect(joining(", "));
```

Ex-8c: Java 8 – Cat & Infl

- Task: find the category and inflection of a term in Lexicon
  1) Find matched records (with matching inflVar)
  2) Map inflVar to category and inflection
  3) Print category and inflection (distinct & sort)

- Example:
  - Input: saw
  - Output: noun|base, noun|sing, verb|base, verb|infinitive, verb|past, verb|pres(fst_plu,second,thr_plur)

Java 8:

```
String outStr = records.parallelStream()
   .filter(rec -> rec.GetInflVar().equals(inTerm))
   .map(rec -> rec.GetCatName() + "|" + rec.GetInflName())
   .distinct()
   .sort()
   .collect(joining(", "));
```

Ex-8d: Java 8 – Cat & Infl by Prefix

- Task: find category and inflection of a term with a given prefix
  1) Find matched records (with matching prefix in inflVar)
  2) Map inflVar to simple format (inflVar|category|inflection|EUI)
  3) Print category and inflection (distinct & sort)

- Example:
  - Input: cashe
  - Output:

    cashed|verb|past_part|E0015477
    cashed|verb|past|E0015477
    cashes|verb|pres(thr_sing)|E0015477
    cashew nuts|noun|plur|E0430044
    cashew nut|noun|base|E0430044
    cashew nut|noun|sing|E0430044
    cashews|noun|plur|E0420703
    cashew|noun|base|E0420703
    cashew|noun|sing|E0420703

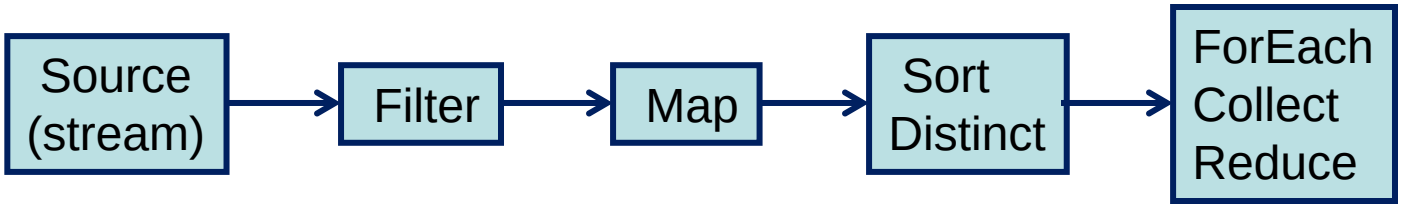
Ex-8d: Java 8 – Cat & Infl Prefix

- Task: find category and inflection of a term with a given prefix
  1) Find matched records (with matching prefix in inflVar)
  2) Map inflVar to category and inflection
  3) Print category and inflection (distinct & sort)

Java 8:

```
String outStr = records.parallelStream()
  .filter(rec -> rec.GetInflVar().startsWith(inTerm))
  .map(rec -> rec.ToSimpleFormat())
  .distinct()
  .sort()
  .collect(joining("\n"));
```

Bulk Data Operation Pattern



Flows Use Inflection tables (17/62)

Flow	Descriptions
-f:b	Uninflect a term
-f:B	Uninflect words in a term
-f:Bn	Normalized uninflect words in a term
-f:Ct	Retrieve citation form
-f:d	Generate derivational variants
-f:e	Generate known uninflected from spelling variants
-f:E	Retrieve the unique EUI for a term
-f:f	Filter output to contain only forms from lexicon
-f:G	Generate all fruitful variants
-f:i	Generate inflectional variants
-f:L	Retrieve category and inflection for a term
-f:Ln	Retrieve category and inflection from lexicon
-f:Lp	Retrieve category and inflection for all terms that begin with the given word
-f:N	Normalize the input text in a non-canonical way (Norm)
-f:R	Generate derivational variants, recursively
-f:s	Generate known spelling variants
-f:U	Convert the output of the Xerox Parc stocastic tagger into lvg style pipe delimited format

Ex-9: Spelling Variants Base

- Task: find all base forms of all spelling variants of a term
  1) Find all EUIs with inflVar matching with inTerm
     - Go through records
     - Filter out records with inflVar not matching inTerm
     - Map matching records to EUI
     - Collect to a set (matchEuiSet)
  2) Find base form that has same EUI in above matchEuiSet
     - Go through records
     - Filter out records that EUI does not matching in matchEuiSet
     - Map matching records to base from
     - Sorted and collect-join with ,

- Example:
  - Input: coloring
  - Algorithm:
    1) E0017902, E0017903
    2) color, colour
  - Output: color, colour

Ex-9: Spelling Variants Base

- Task: find all base forms of all spelling variants of a term

```
List<String> matchEuiSet = records.parallelStream()
  .filter(rec -> rec.GetInflVar().equals(inTerm))
  .map(rec -> GetEui())
  .distinct()
  .collect(toList());

String outStr = records.parallelStream()
  .filter(rec -> MatchEui(matchEuiSet, rec))
  .map(rec -> rec.GetBase())
  .distinct()
  .sort()
  .collect(joining(", "));

Public static boolean MatchEui(List<String> mEuiSet, InflVar rec)
{
  boolean matchFlag = mEuiSet.stream()
    .filter(eEui -> rec.GetEui().equqls(mEui))
    .findFrist()
    .isPresent();
}
```

Ex-9a: Spelling Variants

- Task: find spelling variants of a term
  1) Find matched records (with matching inflVar)
  2) Find inflVar that has same EUI and inflection of matched records
  3) Print inflVar
     - Remove the input term
     - Add , between spelling variants
     - Distinct & sort

- Example:
  - Input: coloring
  - Algorithm:
    1) coloring|verb|pres_part|E0017903|color|color
    2) colouring|verb|pres_part|E0017903|color|color
       ~~coloring|verb|pres_part|E0017903|color|color~~
    3) colouring
  - Output: colouring

Ex-9a: Spelling Variants - Java 7

```
HashSet<InflVar> matchSet = new HashSet<InflVar>();
for(InflVar rec:records)
{
   if(rec.GetInflVar().equals(inTerm) == true)
   {
      matchSet.add(rec);
   }
}
HashSet<String> spVarSet = new HashSet<String>();
for(InflVar rec:records)
{
   for(InflVar matchIv:matchSet)
   {
      if((rec.GetEui().equals(matchIv.GetEui()))
      && (rec.GetInfl().equals(matchIv.GetInfl()))
      && (rec.GetInflVar().equals(matchIv.GetInflVar()) == false))
      {
         spVarSet.add(rec.GetInflVar());
      }
   }
}
ArrayList<String> spVarList = new ArrayList<String>(spVarSet);
Collections.sort(spVarList);
String outStr = new String();
for(int i = 0; i < spVarList.size()-1; i++)
{
   outStr += spVarList.get(i) + ", ";
}
if(spVarList.size() > 0)  // print the last element
{
   outStr += spVarList.get(spVarList.size()-1);
}
```

Ex-9a: Spelling Variants - Java 8

```
List<InflVar> matchSet = records.parallelStream()
    .filter(rec -> rec.GetInflVar().equals(inTerm))
    .distinct()
    .collect(toList());
String outStr = records.parallelStream()
    .filter(rec -> MatchSpVar(matchSet, rec))
    .map(rec -> rec.GetInflVar())
    .distinct()
    .sorted()
    .collect(joining(", "));
private static boolean MatchSpVar(List<InflVar> matchSet, InflVar rec)
{
    boolean matchFlag = matchSet.stream()
        .filter(mRec -> rec.GetEui().equals(mRec.GetEui()))
        .filter(mRec -> rec.GetInfl().equals(mRec.GetInfl()))
        .filter(mRec -> rec.GetInflVar().equals(mRec.GetInflVar()) == false)
        .findFirst()
        .isPresent();
    return matchFlag;
}
```

Ex-9b: Inflectional Variants

- Task: find all base forms of all spelling variants of a term
  1) Find all records with inflVar matching with inTerm (matchSet)
  2) Find all records with matching EUI and base from above matchSet
  3) Print out simple format (inflVar|cat|infl|EUI), distinct & sorted

- Example:
  - Input: resume
  - Output (as simple format):

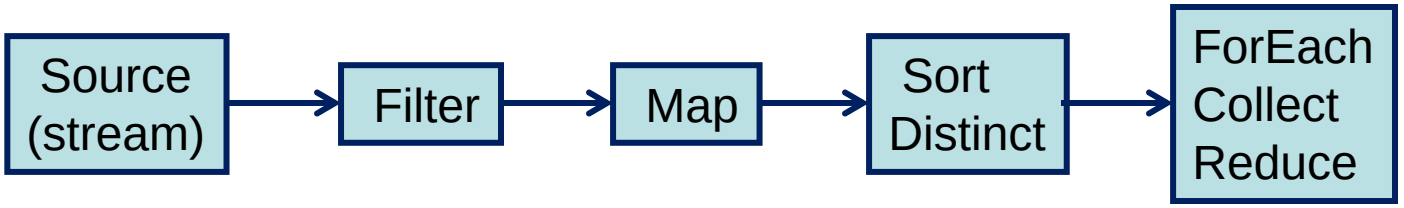
resumed|verb|past_part|E0053098
resumed|verb|past|E0053098
resumes|noun|plur|E0053099
resumes|verb|pres(thr_sing)|E0053098
resume|noun|base|E0053099
resume|noun|sing|E0053099
resume|verb|base|E0053098
resume|verb|infinitive|E0053098
resume|verb|pres(fst_plur,second,thr_plur)|E0053098
resuming|verb|pres_part|E0053098

Ex-9b: Inflectinoal Variants

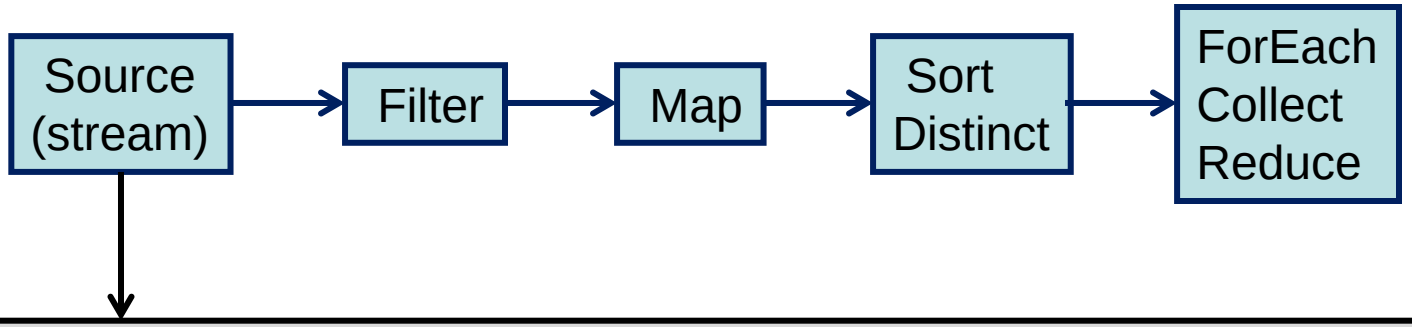
- Task: find all inflectional variants of a term

```
List<InflVar> matchSet = records.parallelStream()
  .filter(rec -> rec.GetInflVar().equals(inTerm))
  .distinct()
  .collect(toList());

String outStr = records.parallelStream()
  .filter(rec -> MatchEuiBase(matchSet, rec))
  .map(rec -> rec.ToSimpleFormat())
  .distinct()
  .sort()
  .collect(joining("\n"));

Public static boolean MatchEuiBase(List<String> mSet, InflVar rec)
{
  boolean matchFlag = mSet.stream()
    .filter(mRec -> rec.GetEui().equqls(mRec.GetEui()))
    .filter(mRec -> rec.GetBase().equqls(mRec.GetBase()))
    .findFrist()
    .isPresent();
    return matchFlag;
}
```

Bulk Data Operation



Java 8 Stream – File Source




```
BufferedReader reader =
    Files.newBufferedReader(Paths.get(inFile), Charset.forName("UTF-8");

reader.lines()
    …
```

Java Source Code Size

	Java 7	Java 8
Ex-1-Iterate	16	13
Ex-2-1-Filter	19	15
Ex-2-1-Filters	20	16
Ex-3-1-Map	21	17
Ex-3-2-uSort	26	19
Ex-3-3-Comparator	41	18
Ex-4-Collector	26	12
Ex-5-CollectToMap	45	29
Ex-6-Reduce	48	27
Ex-7-Contains	13	8
Ex-8-Base	26	10
Ex-9-1-SpVarBase	37	23
Ex-9-2-SpVar	36	24
Ex-9-3-InflVar	38	23

Performance (Time)

	Java 7	Java 8	Java 8 Parallel
Ex-1-Iterate	4.2246	3.2398	3.4801
Ex-2-1-Filter	0.0639	0.0787	0.0320
Ex-2-1-Filters	0.1592	0.1738	0.1034
Ex-3-1-Map	0.1786	0.2461	0.1276
Ex-3-2-uSort	0.3901	0.4770	0.2261
Ex-3-3-Comparator	0.6105	0.8045	0.3457
Ex-4-Collector	0.0907	0.1157	0.0405
Ex-5-CollectToMap	4.0337	5.5560	2.9131
Ex-6-Reduce	1.6480	3.5162	0.9467
Ex-7-Contains	0.0625	0.1400	0.0381
Ex-8-Base	0.0853	0.1353	0.0289
Ex-9-1-SpVarBase	0.2482	0.6405	0.1242
Ex-9-2-SpVar	0.7234	1.2731	0.1754
Ex-9-3-InflVar	0.5625	1.0323	0.1542

Conclusion

Java 8 bulk data operation has:

- better performance with parallelism
- better for code maintenance
  - smaller size
  - readability – control statement
  - writability – syntax easy to learn, good for prototype
- better library APIs: client controls what, library controls how
- software design paradigm is changed (OO to OO/functional)

Future Plan

- Move to Java 8 on internal projects:
  - preprocess and post process (Lexicon release)
- Use Java 8 when developing new code
- LexAccess (2015)
- Lexical Tools (2016 ?)

NLP Applications - Tools

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:
    - Metathesaurus
    - Semantic Network
    - SPECIALIST Lexicon and Lexical Tools

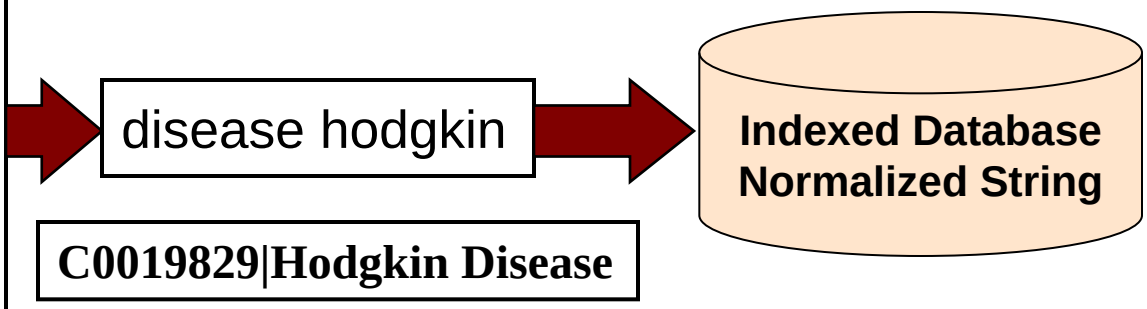
Challenge in Concept Mapping

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

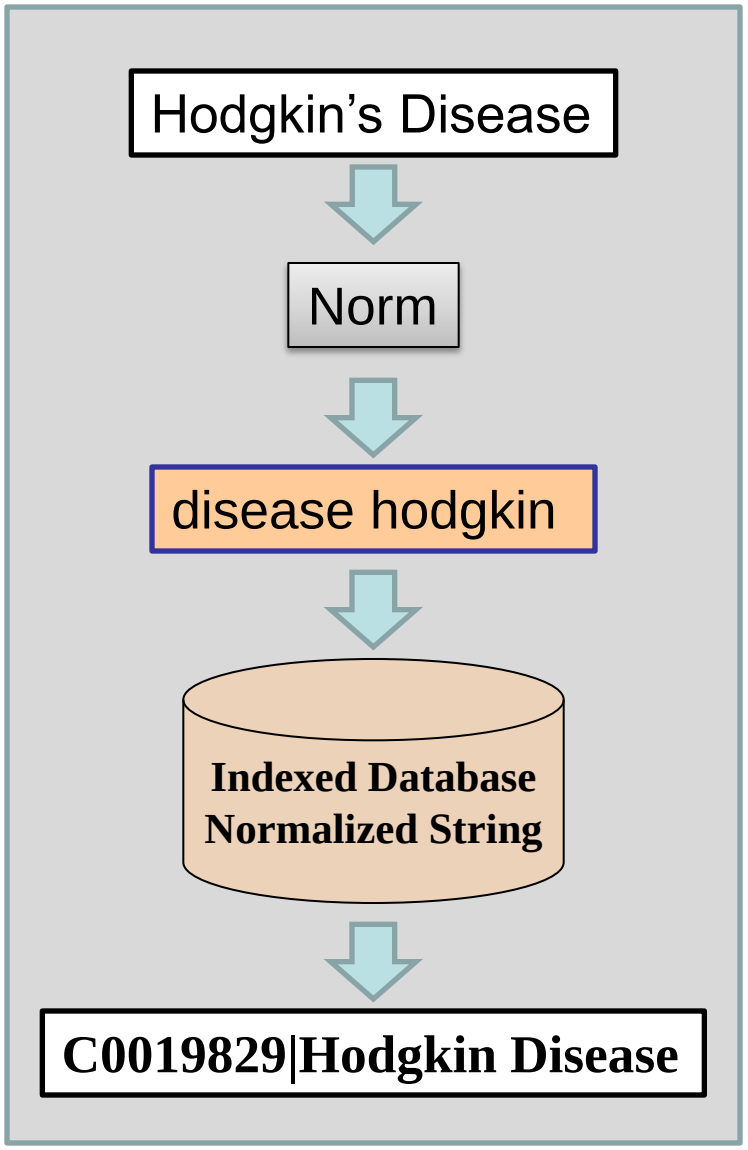
- Concepts have variety of ways to express
  - Example: Hodgkin's Disease
  - ➢ Normalization
  - ➢ Query Expansion

NLP - Norm

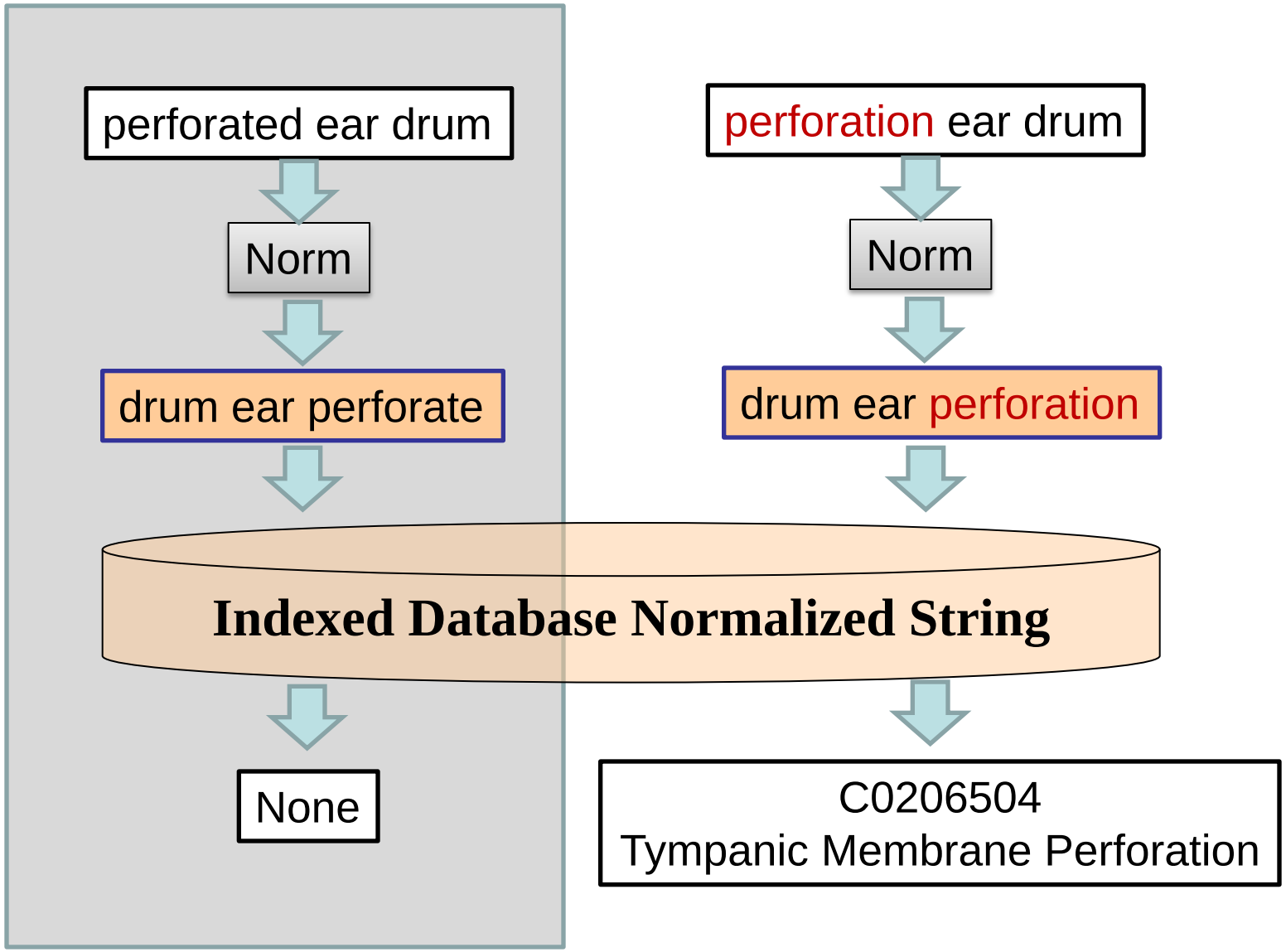
- Hodgkin Disease
- HODGKINS DISEASE
- Hodgkin's Disease
- Disease, Hodgkin's
- HODGKIN'S DISEASE
- Hodgkin's disease
- Hodgkins Disease
- Hodgkin's disease NOS
- Hodgkin's disease, NOS
- Disease, Hodgkins
- Diseases, Hodgkins
- Hodgkins Diseases
- Hodgkins disease
- hodgkin's disease
- Disease;Hodgkins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



NLP - Norm


Hodgkin's Disease
Norm
disease hodgkin
Indexed Database
Normalized String
C0019829|Hodgkin Disease

NLP – Query Expansion



Lexical Tools

- To increase recall & precision

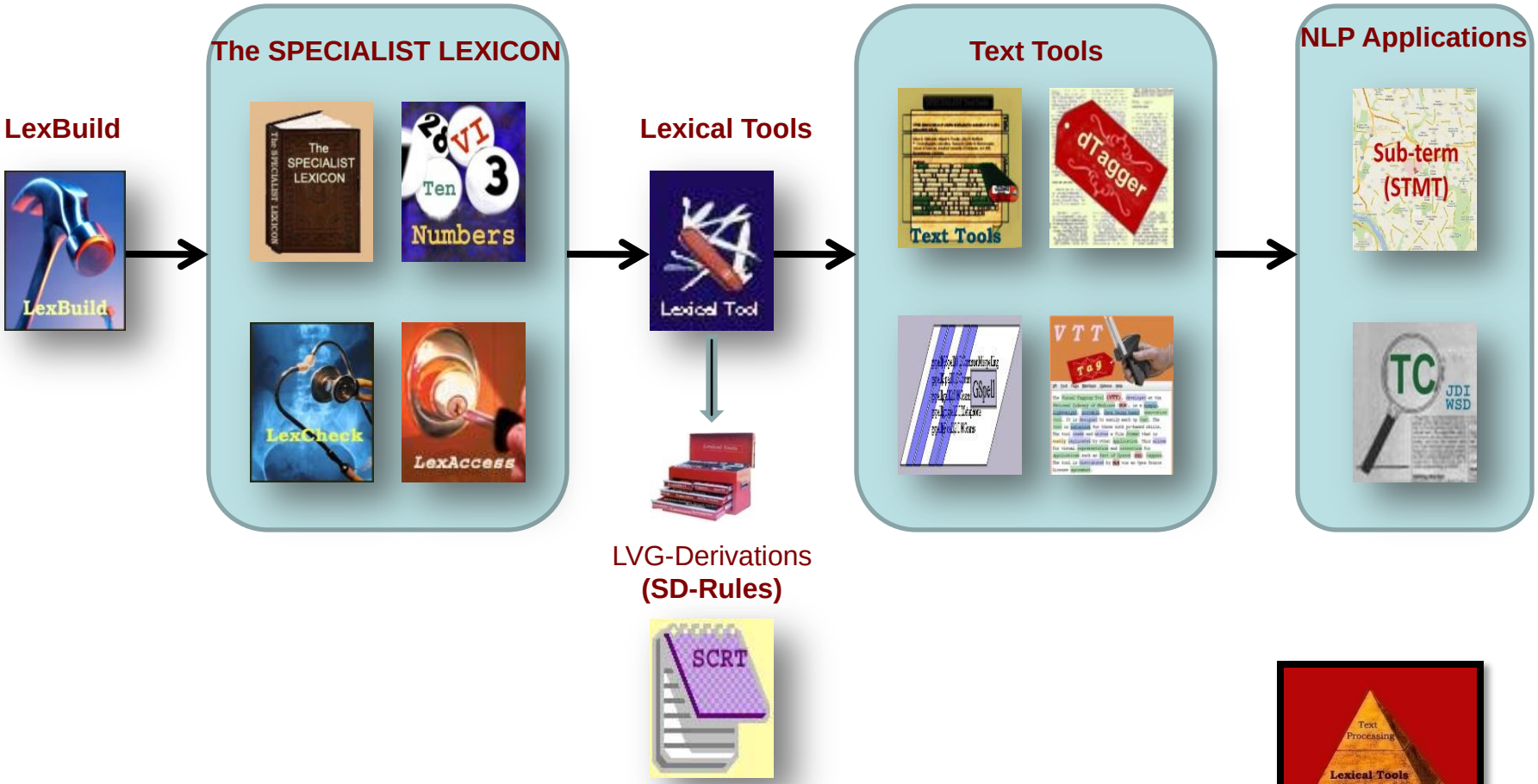
	Query expansion (Recall)	POS Tagging (Precision)
Inputs	perforated ear drum	saw
UMLS-CUI	None	• C1947903|see • C0183089|saw (device)
Process	perforation ear drum	noun
UMLS-CUI	C0206504	C0183089
Preferred term	Tympanic Membrane Perforation	saw (device)

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>

The SPECIALIST NLP Tools



- Lexical Systems Group: http://umlslex.nlm.nih.gov
- The SPECIALIST NLP Tools: http://specialist.nlm.nih.gov

Questions

- Lexical Systems Group: http://umlslex.nlm.nih.gov
- The SPECIALIST NLP Tools: http://specialist.nlm.nih.gov