JoBimText Tutorial NLDB 2015

Practice Session

Martin Riedl & Eugen Ruppert

TU Darmstadt
Language Technology Group

16.06.2015

Outline

1. JoBimViz – JoBimText models in practice
2. Accessing JoBimText models with Java
3. Calculating DT with JoBimText
4. From DT to JoBimText model

Documentation: `http://jobimtext.org`

JoBimViz – JoBimText models in practice – Outline

1. JoBimViz – JoBimText models in practice
   - JoBimText models – A quick recap
   - JoBimViz
2. Accessing JoBimText models with Java
   - IThesaurus Interface
   - WebThesaurus Interface
   - Practice with example project
3. Calculating DT with JoBimText
   - Virtual Machine & Hadoop basics
   - Generating Hadoop script
   - Download and Import of DT
4. From DT to JoBimText model
   - Sense clustering
   - ISA Pattern Extraction
   - Sense Labeling

JoBimText models – Distributional Thesaurus (DT)

- DT is a graph with weighted edges
- self-similarities are included
- top N similar words for a given word
- JoBimText models contain DTs for Jos (terms) and Bims (features)

Jo1	Jo2	Similarity score
mouse	mouse	1000
mouse	Mouse	79
mouse	rat	58
mouse	mice	43

JoBimText models – Jo–Bim scores and counts

- Jo–Bim scores indicate the significance of a Jo–Bim combination,
  based on a significance score (LMI, PMI, LL)
- frequency of the combination is also included
- the Jo–Bim table is usually pruned to remove very frequent (noisy) and infrequent (random) combinations

Jo	Bim	Significance	Count
mouse	Bim_@_Bim(knockout_@_line)	4003.36	247
mouse	Bim_@_Bim(oldfield_@_Thomasomys)	2090.82	129
mouse	Bim_@_Bim(gray_@_lemur)	1475.39	93
mouse	Bim_@_Bim(the_@_cursor)	1321.72	89

JoBimText models – Jo and Bim counts

- Jo and Bim counts help to find out whether a term or feature occurred in the corpus, and how often it occurred
- unpruned data, shows the actual counts

Jo	Count
mouse	18637
mousetrap	253
mouse's	162
mousehole	85

JoBimText models – Sense clusters

- Sense clustering performed by Chinese Whispers algorithm [1]
- sense ID and a list of "cluster terms"
- with ISA labels for the cluster terms

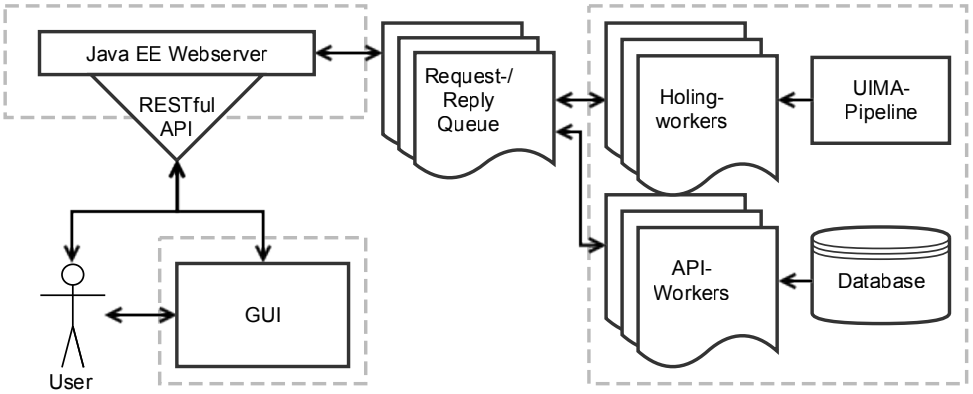
Jo	Sense ID	Cluster terms
mouse	0	musk, mule, roe, barking, ...
mouse	1	mammalian, murine, Drosophila, human, ...
mouse	2	rat, mice, frog, sloth, rodent, ...
...		
mouse	5	joystick, keyboard, monitor, simulation, ...

JoBimViz – Overview

`http://maggie.lt.informatik.tu-darmstadt.de:10080/jobim/`

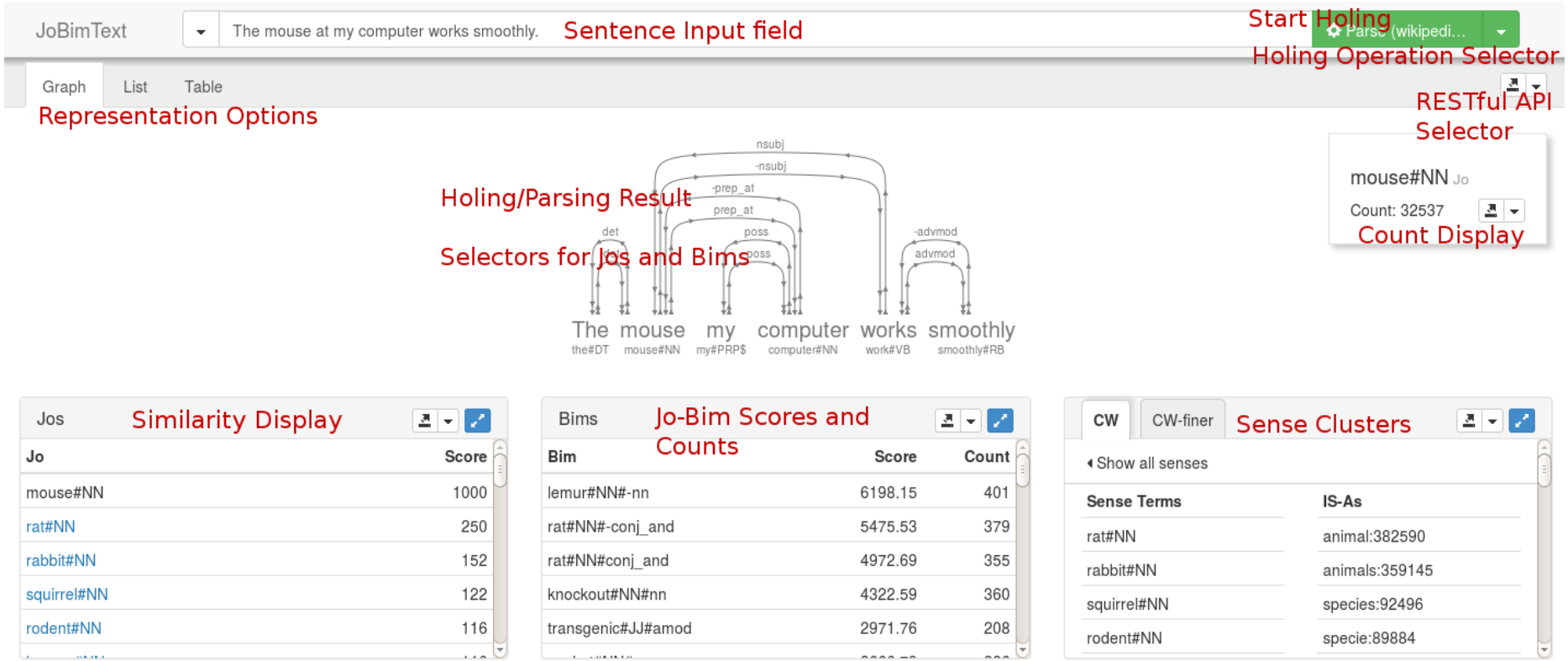
- interactive web application
- offers access to JoBimText models
- sentence holing/parsing
- RESTful API with different output formats:
  JSON, TSV, XML, RDF

JoBimViz – Architecture



- RESTful API
  - abstraction of database operations
  - message queue for robustness
  - useful for prototyping
- GUI
  - uses the RESTful API
  - demonstration of data

JoBimViz Live Demo



http://maggie.lt.informatik.tu-darmstadt.de:10080/jobim/

JoBimViz – Available models

Name	Data	Holing Operation
wikipediaTrigram	Wikipedia EN 2014	Trigram holing
wikipediaStanford	Wikipedia EN 2014	Stanford parsing, dependency holing
trigram	En news 100M	Trigram holing
stanford	En news 100M	Stanford parsing, dependency holing
germanTrigram	De news 70M	Trigram holing
germanParsed	De news 70M	Mate-tools parsing, dependency holing

Identify holing type in URL from a RESTful API request:

`http://maggie.lt.informatik.tu-darmstadt.de:10080/jobim/ws/api/`**`wikipediaTrigram`**`/jo/similar/Passau`

Accessing JoBimText models with Java – Outline

1. JoBimViz – JoBimText models in practice
   - JoBimText models – A quick recap
   - JoBimViz
2. Accessing JoBimText models with Java
   - IThesaurus Interface
   - WebThesaurus Interface
   - Practice with example project
3. Calculating DT with JoBimText
   - Virtual Machine & Hadoop basics
   - Generating Hadoop script
   - Download and Import of DT
4. From DT to JoBimText model
   - Sense clustering
   - ISA Pattern Extraction
   - Sense Labeling

IThesaurus Overview

- general Interface to retrieve data from JoBimText models
- methods for JBT model access
- can access multiple data sources by using different implementations: Database, DCA, DCAlight, JoBimViz
  $\rightarrow$ data sources can be changed without affecting other code
  e.g. prototyping with JoBimViz as data source, then switching to database or DCA for efficiency

IThesaurus Methods

- Counts: `getTermCount(TERM)`, `getContextsCount(CONTEXTS)`
- Similarities: `getSimilarTerms(TERM)`, `getSimilarContexts(CONTEXTS)`, `getSimilarTermScore(TERM, TERM)`
- Term-Contexts counts and scores: `getTermContextsCount(TERM, CONTEXTS)`, `getTermContextsScore(TERM, CONTEXTS)`, `getTermContextsScores(TERM key)`, `getContextsTermScores(CONTEXTS key)`
- Sense clusters and ISAs: `getSenses(TERM)`, `getIsas(TERM)`, `getSenseCUIs(TERM)`

Javadoc API:

http://maggie.lt.informatik.tu-darmstadt.de/jobimtext/doc/org.jobimtext/

IThesaurus configuration and access I

- Construction with a configuration file

```
IThesaurusDatastructure<String, String> dt;

// different data sources possible
dt = new WebThesaurusDatastructure(
        "conf_web_wikipedia_trigram.xml");
dt.connect();

dt = new DatabaseThesaurusDatastructure(
        "conf_mysql_wikipedia_trigram.xml");
dt.connect();
```

IThesaurus configuration and access II

Configuration for JoBimViz:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<webThesaurusConfiguration>
        <protocol>http</protocol>
        <server>maggie.lt.informatik.tu-darmstadt.de</server>
        <port>10080</port>
        <path>/jobim/</path>
        <dataset>wikipediaTrigram</dataset>
</webThesaurusConfiguration>
```

IThesaurus configuration and access III

Configuration for MySQL database (excerpt):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<databaseThesaurusConfiguration>
    <dbUrl>jdbc:mysql://SERVERNAME/wikipedia_trigram?useUnicode=true</dbUrl>
    <dbUser>USER</dbUser>
    <dbPassword>PASSWORD</dbPassword>
    <jdbcString>com.mysql.jdbc.Driver</jdbcString>

    <tables>
        <tableSimilarTerms>LMI_1000_l200</tableSimilarTerms>
        <tableSimilarContexts>LMI_1000_feature_l200</tableSimilarContexts>
       ...
    </tables>

    <similarTermsQuery>select word2, sim FROM $tableSimilarTerms
        WHERE word1=? ORDER BY count desc </similarTermsQuery>
    <similarContextsQuery>SELECT context2, sim FROM $tableSimilarContexts
        WHERE context1 = ? ORDER BY sim desc</similarContextsQuery>
    ...

</databaseThesaurusConfiguration>
```

WebThesaurusInterface vs. IThesaurus interface

- WebThesaurusInterface: same methods as the IThesaurus interface
- realized as access to the RESTful API
- additionally offers sentence holing: transforms sentence into Jos and Bims

```
WebThesaurusInterface<String, String> dtWeb;
dtWeb = new WebThesaurusDatastructure(
        "conf_web_wikipedia_trigram.xml");
dtWeb.connect();

dtWeb.getSentenceHoling("this␣is␣a␣sentence");
```

Result: `[[this,[Bim_@_Bim(_@_is)]], [is,[Bim_@_Bim(this_@_a)]], [a,[Bim_@_Bim(is_@_sentence)]], [sentence,[Bim_@_Bim(a_@_)]]]`

Example Eclipse Project

- example project to access JoBimText models
- demonstration of the available methods
- JoBimViz and MySQL configuration files
- download project from the tutorial page: `https://sites.google.com/site/jobimtexttutorial/resources/`
- unpack and import in Eclipse

IThesaurus – Try it out!

- use the `WebApiStart.java` class as a starting point
- try some tasks, for example:
  1. Determine the term with the highest term count in a sentence!
  2. Find the top 5 similar words for "Passau"!
  3. What are the typical contexts for the term "university"?
  4. Try out different model descriptors (in the examples project)!
- all methods are exemplified in `WebApiExample.java`

IThesaurus – Try it out! – Results

Results for the Wikipedia Trigram model:

1. Term Frequencies for the sentence "this is a sentence":
   a:261,241, is:242,366, this:202,507, sentence:39,239
2. Top 5 similar terms for "Passau":
   Passau, Ingolstadt, Munich, Hildesheim, Bamberg
3. Typical contexts for "university":
   Bim_@_Bim(and_@_professor), Bim_@_Bim(a_@_professor), Bim_@_Bim(public_@_located)

IThesaurus – complex example: contextualization (WSD)

- contextualization, identifying the correct sense in context
- `WebApiContextualizationExample.java` in example project
- Example sentence: "The **mouse** button is stuck"
- Idea:
  - perform sentence holing
  - get the Bim for the target term from the holing output
  - get all senses for the target term
  - for each term from a sense cluster:
    - retrieve the TermContextsCount(term, Bim)
    - add count to results
  - the sense cluster with the highest count is the identified sense in this context

Calculating DT with JoBimText – Outline

1. JoBimViz – JoBimText models in practice
   - JoBimText models – A quick recap
   - JoBimViz
2. Accessing JoBimText models with Java
   - IThesaurus Interface
   - WebThesaurus Interface
   - Practice with example project
3. Calculating DT with JoBimText
   - Virtual Machine & Hadoop basics
   - Generating Hadoop script
   - Download and Import of DT
4. From DT to JoBimText model
   - Sense clustering
   - ISA Pattern Extraction
   - Sense Labeling

VirtualBox VM for Hadoop operations

- VM with preinstalled Hadoop & JoBimText
- Download: `http://sourceforge.net/projects/jobimtextgpl.jobimtext.p/files/hadoop-VM/`
- Requirements:
  - about 5 GB of HD space
  - at least 6GB of memory
  - 64 bit infrastructure
  - installation of VirtualBox: `https://www.virtualbox.org/`

Access to VM

- username: hadoop-user
- password: hadoop (same for root user)
- SSH server configured on port 3022:
  `ssh -p 3022 hadoop-user@127.0.0.1`

Hadoop FS basics

Task	Command
read directory	`hadoop fs -ls`
	`hadoop fs -du [-h]`
create directory	`hadoop fs -mkdir folder`
delete directory	`hadoop fs -rm -r folder`
copy file to HDFS	`hadoop fs -put FILE folder`
delete file	`hadoop fs -rm FILE`
read contents of a folder	`hadoop fs -text folder/*`

http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html

Hadoop FS basics – Practice

- create folder on HDFS
  `hadoop fs -mkdir mouse_corpus`
- download and unpack the mouse_corpus dataset: `wgethttp://maggie.lt.informatik.tu-darmstadt.de/jobimtext/wordpress/wp-content/uploads/2014/04/mouse_corpus.zip`
- upload file from client directly to HDFS
  `hadoop fs -put mouse_corpus mouse_corpus/corpus.txt`
- read text
  `hadoop fs -text mouse_corpus/*`

Generating Hadoop script

- Python script to generate the Hadoop operations shell script
- running the script:
  ```
  cd jobimtext_pipeline_0.1.2/
  python generateHadoopScript.py
  ```
- getting help:
  ```
  python generateHadoopScript.py -h
  ```
- minimal parameters:
  ```
  python generateHadoopScript.py (-hl HOLING | -nh)
  dataset
  ```
  Example DT computation: `python generateHadoopScript.py mouse_corpus -hl matetools_small_lemmatized -f 0 -w 0 -wpfmax 50 -p 100 -l 50 -nb`

Generating Hadoop script – Parameters and options I

Holing options:

- **-hl** set holing operation, e.g. "trigram" or "stanford" selection of holing operations depends on the on the JoBimText pipeline (ASL or GPL)
  The GPL licensed pipeline contains more holing operations (e.g. Stanford dependency holing)
- **-nh** do not perform holing operation; to be used, when using a custom holing operation, or using pre-holed data
- **-savecas** stores the binary CAS after holing operation, useful for parsing large corpora; 'parse once, use often'
- **-lang** document language, set in the CAS; required for Stanford parser and some other DkPro components

Generating Hadoop script – Parameters and options II

JoBimText computation options:

- **-sig** significance measure (LMI, PMI, LL, Freq) for word feature ranking, default: LMI
- **-sc** similarity scoring function of two terms, default: one
  Options:
  - **one** adds a constant '1' for each shared feature
  - **scored** adds $1/|\textit{common terms for feature}|$ for each feature
  - **log_scored** adds $1/\textit{log}(|\textit{common terms for feature}|)$ for each feature

Generating Hadoop script – Parameters and options III

- -af append features to the final DT, default: false
  compiles evidence for similarity score in the DT
- -nb no Bim DT is computed, compute only the Jo DT
- -fm format, for compatibility with former holing functions, v1, v2 or v3

Generating Hadoop script – Parameters and options IV

Pruning options:
(two values can be specified, e.g. -f 2,3; the first value is used for the Jo DT, the second for the Bim DT)

- -f minimal feature count, default: 2
- -w minimal word count, default: 2
- -wf minimal word-feature count, default: 0
- -s minimal significance score, default: 0.0
- -p choose the top p ranked features for each term, default: 1000

Generating Hadoop script – Parameters and options V

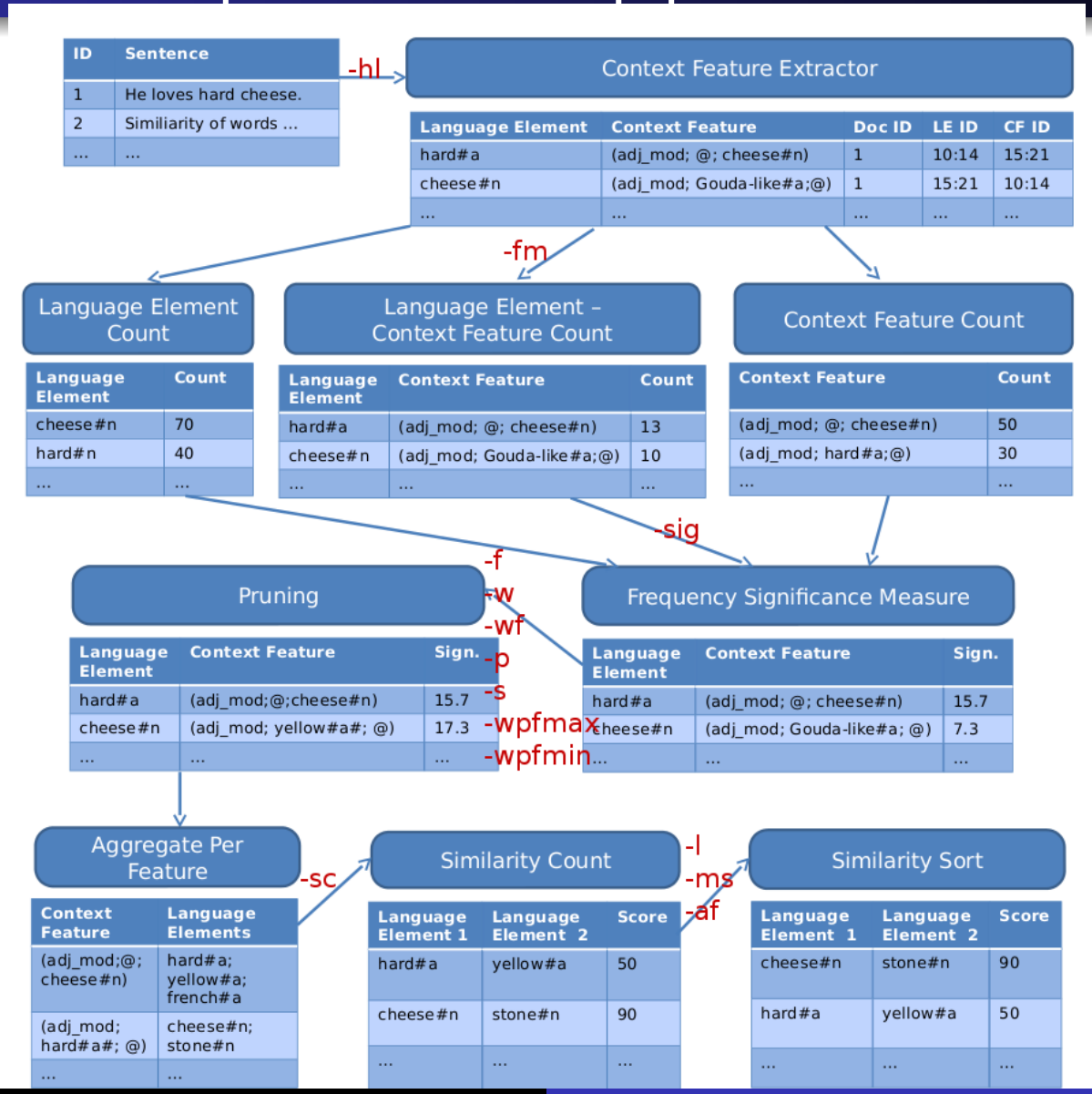
-wpfmax maximal number of words for a feature; features with higher wpf count are discarded, default: 1000

-wpfmin minimal number of words for a feature; features with lower wpf count are discarded, default: 2

-ms minimal similarity, default: 2; removes 'accidental' similarity entries

-l maximal number of similar terms for a term, default: 200

Parameters and options in the pipeline



Distributional Thesaurus – Result data

Which folders to download after computation?

- word count: dataset__WordCount
- feature count: dataset__FeatureCount
- unpruned term-feature scores and counts: dataset__FreqSigLMI
- Jo DT
  - pruned term-feature scores and counts: dataset__FreqSigLMI__PruneContext_SETTINGS
  - similarity graph: dataset__FreqSigLMI__PruneContext_SETTINGS__SimCount_SETTINGS_SimSortlimit
- Bim DT
  - pruned term-feature scores and counts: dataset__FreqSigLMI__PruneContext_BIM_SETTINGS
  - similarity graph: dataset__FreqSigLMI__PruneContext_BIM_SETTINGS__SimCount_SETTINGS_SimSortlimit

DT Computation on Cloud Services

- JoBimText can run on different cloud services, e.g. Amazon EC2
- No need to buy a Hadoop cluster for a single model computation
- Description: `http://maggie.lt.informatik.tu-darmstadt.de/jobimtext/documentation/jobimtext-on-amazon-ec2/`
- Start with the desired Holing operation on a smaller corpus to estimate costs
- Create 'full' DT as the final step

Distributional Thesaurus – DB Import

- use the createTables.py to generate MySQL commands for table creation and data import
- `python createTables.py dataset p sig_measure simsort_limit [path]`

dataset	name of the dataset, e.g. wikipedia_trigram
p	number of features per word, e.g. 1000
sig_measure	significance measure, e.g. LMI
simsort_limit	number of DT entries, e.g. 200
path	optional: absolute path to the dataset folder; when used, import commands will be printed

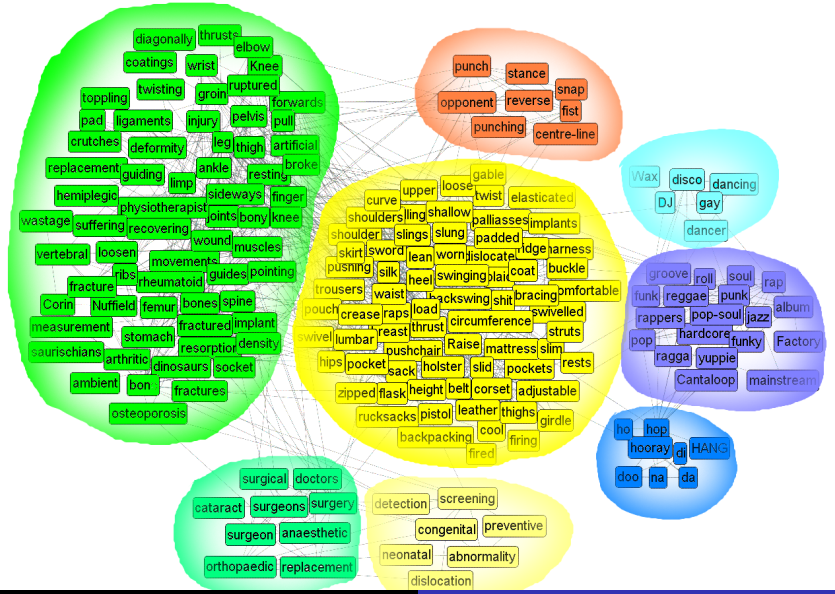
- Example: `python createTables.py wikipedia_mouse 100 LMI 50 /path`

From DT to JoBimText model – Outline

1. JoBimViz – JoBimText models in practice
   - JoBimText models – A quick recap
   - JoBimViz
2. Accessing JoBimText models with Java
   - IThesaurus Interface
   - WebThesaurus Interface
   - Practice with example project
3. Calculating DT with JoBimText
   - Virtual Machine & Hadoop basics
   - Generating Hadoop script
   - Download and Import of DT
4. From DT to JoBimText model
   - Sense clustering
   - ISA Pattern Extraction
   - Sense Labeling

Sense clustering - Overview

- Chinese Whispers, [1]
- unsupervised graph clustering algorithm
- Word Sense Induction (word sense clustering)
- Documentation:
  `http://maggie.lt.informatik.tu-darmstadt.de/jobimtext/components/chinese-whispers/`



Sense clustering – input/output data

- input: DT file
  - format: (word1, word2, similarity)
  - example DT: mouse_corpus_dt
- output:
  - format: (word, sense_id, list-of-sense-terms)
  - example sense clustering:

    mouse 0 cat,dog,rat
    mouse 1 keyboard,joystick
  - example file `mouse_corpus_senses`

Download the JoBimText pipeline for execution:

`https://sourceforge.net/projects/jobimtext/files/latest/download`

Sense clustering – Options and arguments

- required arguments:

  -i input file
  -o output file

- optional arguments

  -a weighting algorithm: 1=constant, lin=linear or log=logarithmic (default: 1)
  -N number of top DT entries to consider (default: MAX)
  -n number of top edges to consider within entry (default: MAX)
  -ms minimal similarity (default: 1)
  -mr maximal cluster rank (default: MAX)
  -mc minimal cluster size (default: 1)

Sense clustering – Execution

Recommended General Settings:

```
java -cp lib/org.jobimtext-0.1.2.jar:lib/*
org.jobimtext.sense.ComputeSenseClusters -a 1 -N 200
-n 100 -mc 3 -ms 5 -mr 100 -i DT_FILE -o OUTPUT_FILE
```

Example Settings:

```
java -cp lib/org.jobimtext-0.1.2.jar:lib/*
org.jobimtext.sense.ComputeSenseClusters -N 50 -n 50
-i mouse_corpus_dt -o mouse_corpus_senses
```

ISA Pattern Extraction

PattaMaika
`http://maggie.lt.informatik.tu-darmstadt.de/jobimtext/components/pattamaika/`

- UIMA pipeline (OpenNLP components)
- UIMA RUTA for pattern identification
- Hearst Patterns in RUTA ([2] and [3]):

```
(_NP (COMMA _NP)* ("and" | "or")  "other" _NP{->TEMP})
 {-PARTOF(PATTERN)-> CREATE(PATTERN,"x"=TEMP)};
```

- Matches "She likes *cats*, *dogs* and other *animals*"

Running PattaMaika – Hadoop

- Hadoop shell script creation:
  `python generatePattamaikaHadoopScript.py dataset [-q queue-name]`
- execution by running the shell script
- detailed instructions:
  `http://maggie.lt.informatik.tu-darmstadt.de/jobimtext/documentation/pattern-extraction-with-pattamaika/`

Running PattaMaika – Local

- create 'corpus' folder:
  `mkdir corpus`
- copy corpus to 'corpus' folder: `cp /path/to/mouse_corpus corpus/mouse.txt`
- execution by running the PattaMaika descriptor
  ```
  java -Xmx3g -cp "lib/*"
  org.jobimtext.util.RunJoBimIngestionLocal
  descriptors/PattamaikaUIMAOperations.xml
  ```
- results are stored in 'pattern_out' folder

PattaMaika – Results

- wikipedia_1M dataset: ca. 200,000 patterns, 6,000 with frequency > 1
- most frequent ISA patterns:

Pattern	Frequency
English ISA language	27
English ISA languages	26
China ISA countries	23
China ISA country	23
Australia ISA countries	22
Australia ISA country	22
Canada ISA countries	21
Canada ISA country	21
United_States ISA countries	19
United_States ISA country	19
India ISA country	18

PattaMaika – Results from Example corpus

- mouse_corpus dataset: 181 patterns, 8 with frequency > 1
- most frequent ISA patterns (see mouse_corpus_patterns):

Pattern	Frequency
fish ISA animals	4
fish ISA animal	4
fish ISA aquatic_animals	3
fish ISA aquatic_animal	3
crustacean ISA fish	2
fish ISA organism	2
fish ISA organisms	2
crustaceans ISA fish	2

Sense Labeling – Overview

- ISA labeling of sense clusters
- Input data: ISA patterns, sense clusters
- Execution:

```
java -cp lib/org.jobimtext.pattamaika-0.1.2.jar:lib/*
org.jobimtext.pattamaika.SenseLabeller -mf 1 -ms 2 -mm 1
-sep "#" -tsep ', ' -p pattern_out/pattern_out_0.txt -s
mouse_senses -o mouse_senses_isa
```

- detailed instructions and examples:
  http://maggie.lt.informatik.tu-darmstadt.de/jobimtext/documentation/sense-labelling/

Sense Labeling – Input and output data

- patterns:

mouse ISA animal	15
cat ISA animal	10
dog ISA animal	20
dog ISA pet	5

- sense cluster:

mouse	0	cat,dog,rat
mouse	1	keyboard,joystick

- result:

mouse	0	cat,dog,rat	animal:60, pet:5
mouse	1	keyboard,joystick	product:20, input_device:2

Thank you!

Thank you for your attention! Good efforts with JoBimText models!

Questions? Comments?

References

- Biemann, C. (2006). Chinese whispers – an efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of TextGraphs: The First Workshop on Graph Based Methods for Natural Language Processing*, New York City, NY, USA, pp. 73–80.
- Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *Proc. COLING-1992*, Nantes, France, pp. 539–545.
- Klaussner, C. and D. Zhekova (2011). Lexico-syntactic patterns for automatic ontology building. In *RANLP Student Research Workshop*.