EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN


# GermaNet Workshop Part 3: Programmatical Access to GermaNet

**Verena Henrich, Düsseldorf, 19. Februar 2015**

# The GermaNet Java APIs

- There are two Java APIs for GermaNet:

  1) “General“ GermaNet Java API for accessing all GermaNet data

  2) Semantic relatedness API for calculating semantic relatedness

# The “general“ Java API for accessing GermaNet

- The Java API to GermaNet represents a programming interface, i.e., it provides methods to access GermaNet programmatically
- It is intended to be a read-only resource, no methods to extend or modify data are provided
- The API reflects the GermaNet structure:
  - There are classes for representing lexical units, synsets, compounds, examples sentences, verbal frames, Wiktionary paraphrases, interlingual links to the Princeton WordNet, etc.
  - There are enums encoding word classes, semantic fields, lexical and conceptual relations, compound properties, etc.
- All classes and methods are described in the Javadoc: http://www.sfs.uni-tuebingen.de/GermaNet/documents/api/javadoc9.0/index.html

# Using the GermaNet Java API

- Before we start, we need access to
  - The GermaNet XML data: folder **`"GN_V90_XML"`**
  - The GermaNet Java API called **`GermaNetApi9.0.1.jar`**: http://www.sfs.uni-tuebingen.de/GermaNet/tools.shtml#APIs

- Add **`GermaNetApi9.0.1.jar`** to the classpath of your project
- Increase the allocated memory by using the following virtual machine options: **`-Xms256m -Xmx256m`**

# First Step: Creating a GermaNet Object

- Include the GermaNet API library:
  ```
  import de.tuebingen.uni.sfs.germanet.api.*;
  ```
- Main class is called **`GermaNet`** and serves as a starting point
- When a **`GermaNet`** object is constructed, the GermaNet data is loaded from XML files
- The call to the constructor might throw an exception: we need exception handling to prevent the case that the files are not found

# First Step: Creating a GermaNet Object

- Constructor loads GermaNet data from a directory which is specified by a parameter
  - Either as a **`String`** object:

```
GermaNet gnet = new GermaNet("GN_V90_XML");
```

  - Or as a **`File`** object:

```
File gnetDir = new File("GN_V90_XML");
GermaNet gnet = new GermaNet(gnetDir);
```

- Optional **`ignoreCase`** parameter:

```
GermaNet gnet = new GermaNet("GN_V90_XML", true);

GermaNet gnet = new GermaNet(gnetDir, true);
```

# Getting Synsets from a GermaNet Object

- The **`GermaNet`** object provides methods for retrieving lists of synsets (class **`Synset`**) and lexical units (class **`LexUnit`**), which can be filtered by e.g., word category or orthographic form
- Retrieve list of all **`Synsets`**:

```
List<Synset> allSynsets = gnet.getSynsets();
```

- Retrieve list of all **`Synsets`** containing “Bank”:

```
List<Synset> bankSynsets = gnet.getSynsets("Bank");
```

- Retrieve list of all **`Synsets`** with word category adjective:

```
List<Synset> adjSynsets =
                    gnet.getSynsets(WordCategory.adj);
```

# Working with Synsets (Extract Paraphrase and WordCategory)

- A **`Synset`** object has methods for retrieving its word category, lexical units, and paraphrases, as well as methods for retrieving synsets that are related to it.

- Retrieving the paraphrase:

```
String paraphrase = aSynset.getParaphrase();
```

- To get a **`Synset`**'s word category and do further processing in case of an adjective:

```
WordCategory wc = aSynset.getWordCategory();
if (wc == WordCategory.adj) {
    // do something
}
```

# Working with Synsets (Extract LexUnits and OrthForms)

- To get a list of all **`LexUnits`** of a **`Synset`** and iterate through them:

```
List<LexUnit> lexList = aSynset.getLexUnits();
for (LexUnit lu : lexList) {
    // process lexical unit
}
```

- To get a **`Synset`**'s orthographic forms (retrieves a list of all orthographic forms in all the **`LexUnits`** that belong to this **`Synset`**):

```
List<String> orthForms =
                        aSynset.getAllOrthForms();
```

# Working with Synsets (Extract Related Synsets)

- To extract a list of all **`Synsets`** that are related to **`aSynset`**:

```
List<Synset> allRelations =
                          aSynset.getRelatedSynsets();
```

- Suppose you want to find all hypernyms of a **`Synset`**:

```
List<Synset> hypernyms =
   aSynset.getRelatedSynsets(ConRel.has_hypernym);
```

- Suppose you want to find all hyponyms of a **`Synset`**:

```
List<Synset> hyponyms =
   aSynset.getRelatedSynsets(ConRel.has_hyponym);
```

- See the Javadoc for more values of **`ConRel.xxx`**

# Getting LexUnits from a GermaNet Object

- Works analogously to extracting **`Synsets`**
- Retrieve list of all **`LexUnits`**:

```
List<LexUnit> allLexUnits = gnet.getLexUnits();
```

- Retrieve list of all **`LexUnits`** containing “Bank”:

```
List<LexUnit> bankLexUnits =
                        gnet.getLexUnits(“Bank“);
```

- Retrieve list of all **`LexUnits`** with word category noun:

```
List<LexUnit> nomLexUnits =
                gnet.getLexUnits(WordCategory.nomen);
```

# Working with LexUnits (Extract OrthForm, Check Named Entity & Artificial)

- To retrieve the main orthographic form of a lexical unit:

```
String orthForm = aLexUnit.getOrthForm();
```

- To check whether a lexical unit is a named entity:

```
boolean isNamedEntity = aLexUnit.isNamedEntity();
```

- To check whether a lexical unit is artificial:

```
boolean isArtificial = aLexUnit.isArtificial();
```

- See the Javadoc for a complete list of methods

# Working with LexUnits (Extract Related LexUnits)

- To extract a list of all **`LexUnits`** that are related to **`aLexUnit`**:

```
List<LexUnit> allRelations =
                aLexUnit.getRelatedLexUnit();
```

- Suppose you want to find all antonyms of a **`LexUnit`**:

```
List<LexUnit> antonyms =
  aLexUnit.getRelatedLexUnit(LexRel.has_antonym);
```

- See the Javadoc for more values of **`LexRel.xxx`**

# Two Ways of Extracting Synonyms – Solution 1

- Suppose you want to find all synonyms of “Medikament” (note that there is only one reading of “Medikament” in GermaNet)

- First solution:
  1. Extract the one **`LexUnit`** representing “Medikament”:

```
LexUnit medLU =
          gnet.getLexUnits(“Medikament“).get(0);
```

  2. Extract synonyms of **`medLU`**:

```
List<LexUnit> synonyms =
   medLU.getRelatedLexUnits(LexRel.has_synonym);
```

# Two Ways of Extracting Synonyms – Solution 2

- Second solution:
  1. Extract the one **`Synset`** representing “Medikament”:

```
Synset medSynset =
            gnet.getSynsets(“Medikament“).get(0);
```

  2. Extract **`LexUnits`** of **`medSynset`**:

```
List<LexUnit> synonyms =
                        medSynset.getLexUnits();
```

- Note that the list of synonyms differ in the two solutions: The list in solution 1 contains the synonymous **`LexUnits`** only, whereas the list in solution 2 also contains the **`LexUnit`** representing “Medikament”.

# Exercise 1: How many readings have “Dollar“, “haben“, and “sauer”, respectively?

# Exercise 1: How many readings have “Dollar“, “haben“, and “sauer”, respectively?

```
List<LexUnit> dollarLUs = gnet.getLexUnits("Dollar");
System.out.println("\"Dollar\": " + dollarLUs.size());
```

→ `"Dollar": 15`

```
List<LexUnit> schauenLUs = gnet.getLexUnits("schauen");
System.out.println("\"schauen\": " + schauenLUs.size());
```

→ `"schauen": 5`

```
List<LexUnit> sauerLUs = gnet.getLexUnits("sauer");
System.out.println("\"sauer\": " + sauerLUs.size());
```

→ `"sauer": 2`

# Exercise 2: Extract all nominal LexUnits, but without named entities and without artificial nouns

# Exercise 2: Extract all nominal LexUnits, but without named entities and without artificial nouns

```
List<LexUnit> lexList = new ArrayList<LexUnit>();

for (LexUnit lu : gnet.getLexUnits(WordCategory.nomen)) {
    if (!lu.isNamedEntity() && !lu.isArtificial())
        lexList.add(lu);
}
```

# Exercise 3: Identify 10 Synsets in GermaNet that contain at least 10 synonymous LexUnits

# Exercise 3: Identify 10 Synsets in GermaNet that contain at least 10 synonymous LexUnits

```
List<Synset> atLeast10LexUnits = new ArrayList<Synset>();

for (Synset synset : gnet.getSynsets()) {
    if (atLeast10LexUnits.size() >= 10)
        break;

    if (!atLeast10LexUnits.contains(synset)
        && synset.getLexUnits().size() >= 10)
        atLeast10LexUnits.add(synset);
}

System.out.println("10 synsets with at least 10 ”
                                    + "synonymous LUs: ");

for (Synset s : atLeast10LexUnits)
    System.out.println(s.getAllOrthForms() + s.getId());
```

# Exercise 4: Find 8 words in GermaNet that show at least 8 readings

# Exercise 4: Find 8 words in GermaNet that show at least 8 readings

```
List<String> atLeast8Readings = new ArrayList<String>();

for (LexUnit lu : gnet.getLexUnits()) {
    if (atLeast8Readings.size() >= 8)
        break;

    if(!atLeast8Readings.contains(lu.getOrthForm())
       && gnet.getLexUnits(lu.getOrthForm()).size() >= 8)
       atLeast8Readings.add(lu.getOrthForm());
}

System.out.println("8 words with at least 8 readings: "
                                        + atLeast8Readings);
```

# Exercise 5: Try to find 5 synsets that have exactly 5 hyponyms

# Exercise 5: Try to find 5 synsets that have exactly 5 hyponyms

```
List<Synset> exactly5Hyponyms = new ArrayList<Synset>();

for (Synset s : gnet.getSynsets()) {
    if (exactly5Hyponyms.size() >= 5)
        break;

    if(!exactly5Hyponyms.contains(s) &&
       s.getRelatedSynsets(ConRel.has_hyponym).size() == 5)
       exactly5Hyponyms.add(s);
}

System.out.println("5 synsets with exactly 5 hyponyms: ");

for (Synset s : exactly5Hyponyms)
    System.out.println(s.getAllOrthForms() + s.getId());
```

# Exercise 6: Extract related Synsets of a given Synset, where the relation type is provided as a String

- Sometimes you may have a conceptual relationship represented as a `String`.

# Exercise 6: Extract related Synsets of a given Synset, where the relation type is provided as a String

- Sometimes you may have a conceptual relationship represented as a **`String`**. The following code can be used to validate the **`String`** and retrieve the relations of **`aSynset`**:

```
String aRel = "has_hypernym";
List<Synset> relList;

// make sure aRel is a valid conceptual relation
if (ConRel.isConRel(aRel)) {
    relList = aSynset.getRelatedSynsets(
                                ConRel.valueOf(aRel));
}
```

# Exercise 7: Recursively print the hypernymy hierarchy of all artificial adjectives (i.e., subclasses)

# Exercise 7: Recursively print the hypernymy hierarchy of all artificial adjectives (i.e., subclasses)

```
Synset rootSynset = gnet.getSynsets("GNROOT").get(0);
printSubclasses(rootSynset, 0);

public void printSubclasses(Synset syn, int depth) {
    for (Synset s : syn.getRelatedSynsets(ConRel.has_hyponym)) {
        if (s.inWordCategory(WordCategory.adj)
            && s.getLexUnits().get(0).isArtificial()) {

            for (int i = 0; i < depth; i++)
                System.out.print(" > ");

            System.out.println(s.getAllOrthForms().get(0));
            printSubclasses(s, depth + 1);
        }
    }

    if (depth == 1)
        System.out.println("");
}
```

# Semantic Relatedness API

- The semantic relatedness API is also implemented in Java
- Available at: http://www.sfs.uni-tuebingen.de/GermaNet/tools.shtml#SemRelAPI
- It can be used for calculating semantic relatedness between any two words/lexical units in GermaNet
- Implemented relatedness algorithms include:
  - Hirst and St-Onge, 1998
  - Jiang and Conrath, 1997
  - Leacock and Chodorow, 1998
  - Lesk, 1986
  - Lin, 1998
  - Resnik, 1995
  - Wu and Palmer, 1994

# Semantic Relatedness API

- The download package includes a basic graphical user interface
- All classes and methods are described in the Javadoc:
  http://www.sfs.uni-tuebingen.de/GermaNet/documents/api/javadocSemRel8.0/index.html
- Class `Relatedness` is the starting point to the API
- Also see the tutorial that comes with the download package for further information

# Creating the Frequency File

- For algorithms that rely on information content, frequency information is required
- Before using the Semantic Relatedness API you need to create a frequency file
- Class **`Frequency`** provides a static method to create this file:
  - You need to specify the output folder (i.e. **`frequencies`** in the example)

```
GermaNet gnet = new GermaNet("GN_V90_XML");
Frequency.assignFrequencies("frequencies", gnet);
```

- After this file has been created you can either use the API or the standalone GUI

# Using the Semantic Relatedness API

- Class **`Relatedness`** is the starting point to the API

```
Relatedness rel = new Relatedness(gnet);
```

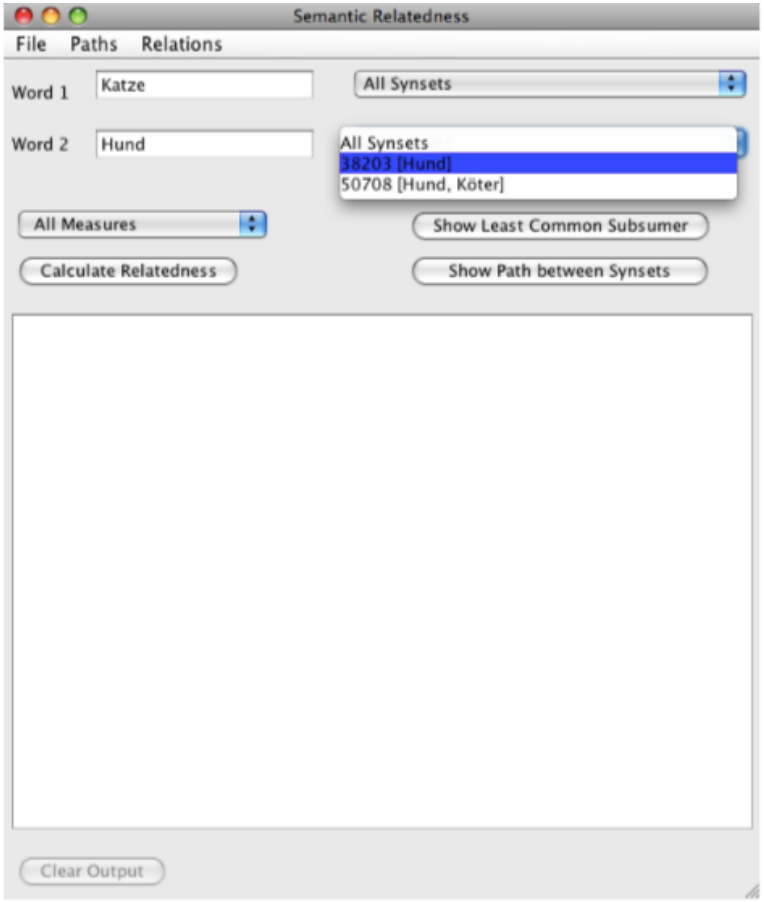
- Important note: the file **`relatedness.ini`** needs to be available in the current folder
- Class **`Relatedness`** includes methods for several relatedness algorithms

```
Synset syn1 = gnet.getSynsets("Bank").get(0);
Synset syn2 = gnet.getSynsets("Flugzeug").get(0);
rel.wup(syn1, syn2);
rel.lin(syn1, syn2);
rel.hirstAndStOnge(syn1, syn2);
rel.lesk(syn1, syn2);
```

- See the Javadoc for more values of Relatedness.xxx

# Using the Semantic Relatedness GUI

- Class **`MainFrame`** is the starting point to the GUI
- Important: file **`relatedness.ini`** has to be in the same folder as the **`jar`** file
- The command
  **`java -jar SemanticRelatedness GN_V90_XML frequencies`**
  will start the GUI.
- Once started you can enter words between which you want to calculate semantic relatedness



# Semantic Relatedness References


- Hirst, G. and D. St-Onge: Lexical Chains as Representations of Context for the Detection and Correction of Malapropisms. In C. Fellbaum (ed.), WordNet: An Electronic Lexical Database, The MIT Press, 1998.
- Jiang, J.J. and D.W. Conrath: Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In Proceedings of ROCLING 1997, Taiwan, 1997.
- Leacock, C. and M. Chodorow: Combining Local Context and WordNet Similarity for Word Sense Identification. In C. Fellbaum (ed.), WordNet: An Electronic Lexical Database, The MIT Press, 1998.
- Lesk, M.: Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. Proceedings of SIGDOC ’86, New York, NY, USA, 1986.
- Lin, D.: An Information-Theoretic Definition of Similarity. Proceedings of ICML ’98, San Francisco, CA, USA, 1998.
- Resnik, P.: Using Information Content to Evaluate Semantic Similarity in a Taxonomy. Proceedings of the 14th international joint conference on Artificial intelligence - Volume 1, San Francisco, CA, USA, 1995.
- Wu, Z. and M. Palmer: Verb Semantics And Lexical Selection. Proceedings of ACL ’94, Stroudsburg, PA, USA, 1994.