Content Quality Management with Open Source Language Technology

Christian Lieske (SAP AG)
Dr. Felix Sasaki (DFKI, FH Potsdam)

To complement this presentation, the full text for the conference proceedings has been included in the appendix to this presentation.

tcWorld 2011 – Wiesbaden

October 20, 2011

SAP

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

tcworld conference 2011

Authors

Prof. Dr. Felix Sasaki

DFKI/FH Potsdam

- Appointed to Prof. in 2009; since 2010 senior researcher at DFKI (LT-Lab)
- Head of the German-Austrian W3C-Office
- Before, staff of the World Wide Web Consortium (W3C) in Japan
- Main field of interest: combined application of W3C technologies for representation and processing of multilingual information
- Studied Japanese, Linguistics and Web technologies at various Universities in Germany and Japan

Christian Lieske

Globalization Services
SAP AG

- Knowledge Architect
- Content engineering and process automation (including evaluation, prototyping and piloting)
- Main field of interest: Internationalization, translation approaches and natural language processing
- Contributor to standardization at World Wide Web Consortium (W3C), OASIS and elsewhere
- Degree in Computer Science with focus on Natural Language Processing and Artificial Intelligence

Expectations ?

You expect …

Demonstration of a specific format, solution, method or procedure in practice.

An offering related to technical authoring

Get a handle on Content Quality Management

A quiet place far away from the fair ;-)

A tutorial designed for the professional level audience

Interaction

A possibility to talk about your challenges and approaches

Basics of Language Technology

That‘s what we expected …

Overview

Linguistic Quality Management and Control for Textual Content

Basics of Natural Language Processing (NLP)/Language Technology (LT)

Text Technology

NLP/LT and Text Technology in real world deployment scenarios

NLP/LT and Text Technology in the Open Source world

- **Remarks on Basic Due Diligence related to Open Source**
- **Okapi/Rainbow/CheckMate**
- **LanguageTool**
- **Apache Solr/Apache Lucene**
- **Unstructured Information Management Architecture**

Conclusion and Outlook

WARM UP

Dimensions of Content Quality

```
<p>We are presenting on the <a hrif="http://www.tekom.de/">tcworld conference 2011</a>
<img src="tcWorld2011.jpg" width="450" height="174" alt="The tcWorld 2011 Logo"/>.</p>
```

Content is more than text

Quality Management is more than Quality Control

Often more than just linguistic stuff is in the mix (Natural Language Processing vs. Text Technology)

Dimensions of Tasks related to Linguistics

Is there existing or new terminology?

Are spelling, grammar, and style alright?

Can I recycle an existing translation?

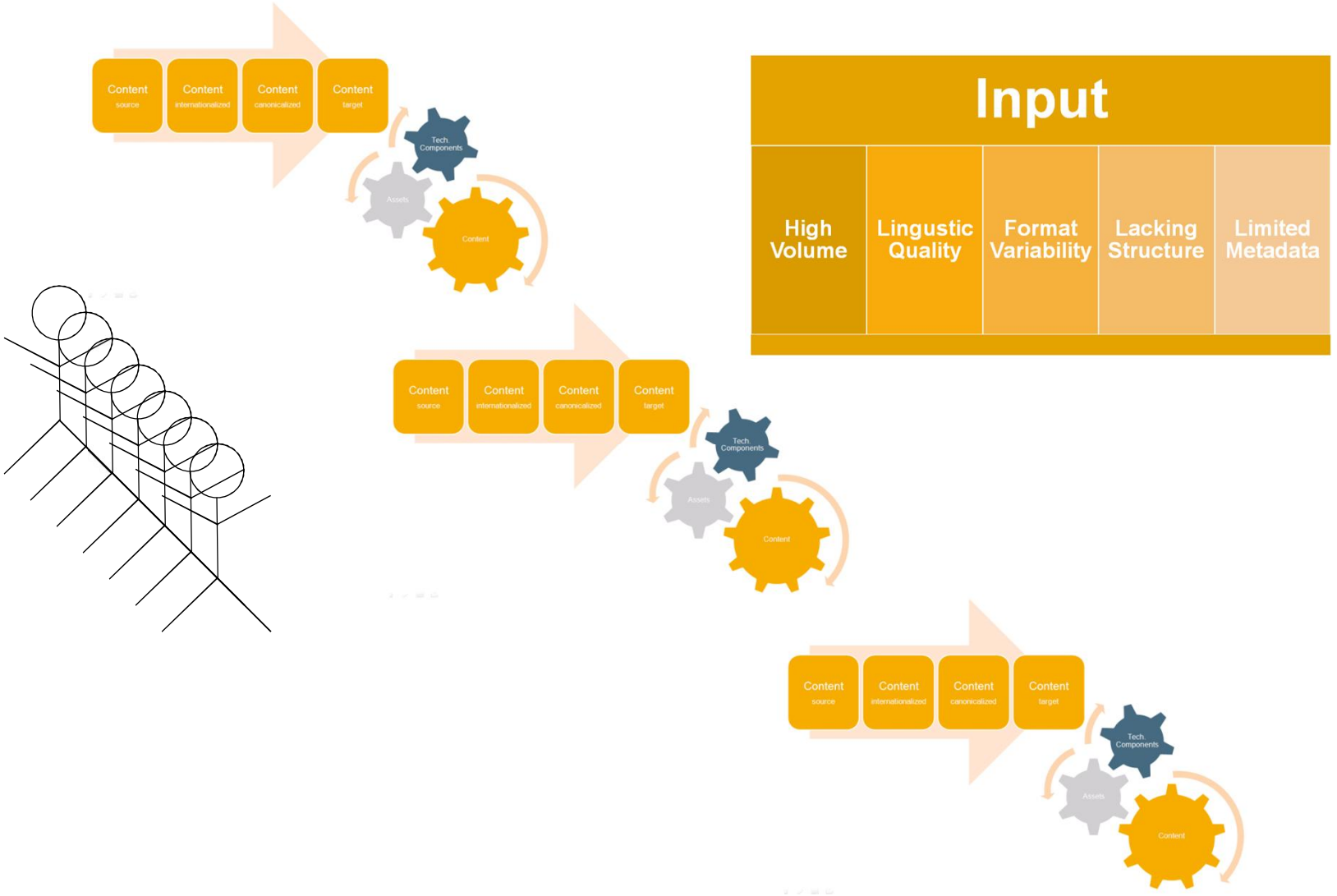
Should I insert markup of some kind?

```
<p>Text technology provides answers not only related to characters, but also to other areas:</p>
<ul>
  <li>Content (e.g. HTML, XML, XML-based vocabularies like DocBook or DITA, ...)</li>
  <li>Metadata (e.g. RDF)</li>
  <li>Filters to go from general XML to XLIFF via ITS</li>
</ul>
```

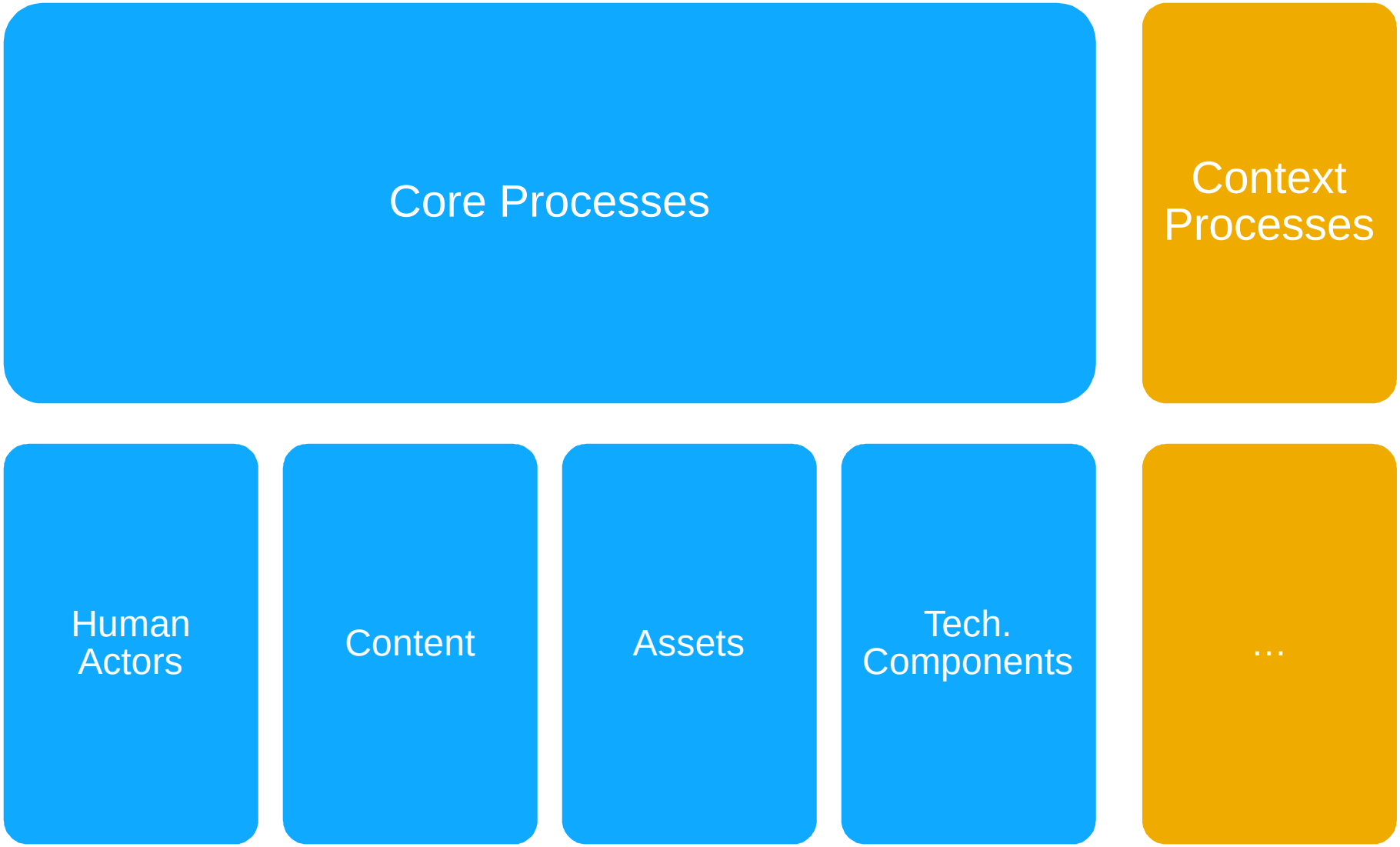
```
<p>テキストテクノロジーは文字だけではなく、下記の他の分野にも基本になる。</p>
```

```
<p>For further inquiries, please contact our office at <span class="tel">+49 331 967675</span>.</p>
```

Challenge of the Real World (1/3)



Challenge of the Real World (2/3)



Challenge of the Real World (3/3)

Anyone, anything (proprietary, XML ...), anytime

Scaling, consistency, compliance …

Coupling

- Object Linking and Embedding, HTTP, Web Services, ...
- Libraries/Application Programming Interfaces/Software Development Kits
- Orchestration (e.g. synchronization of calls, and "bus-like" integration or annotation framework)

Adressing the Challenges (1/2)

Best Practices and Standardization

Computer-Assisted Linguistic Quality Support

Computer-Assisted Linguistic Assistance

i. Needs assets
ii. Creates assets
iii. Relates to Natural Language Processing

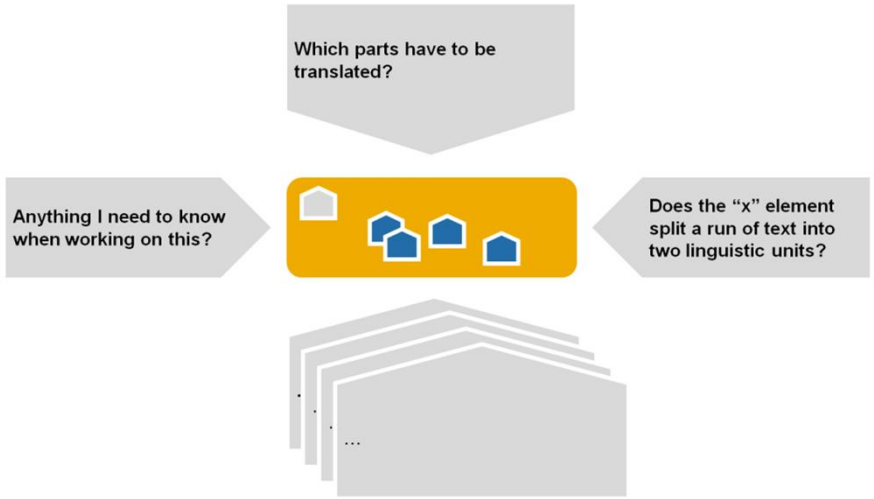
Adressing the Challenges (2/2)

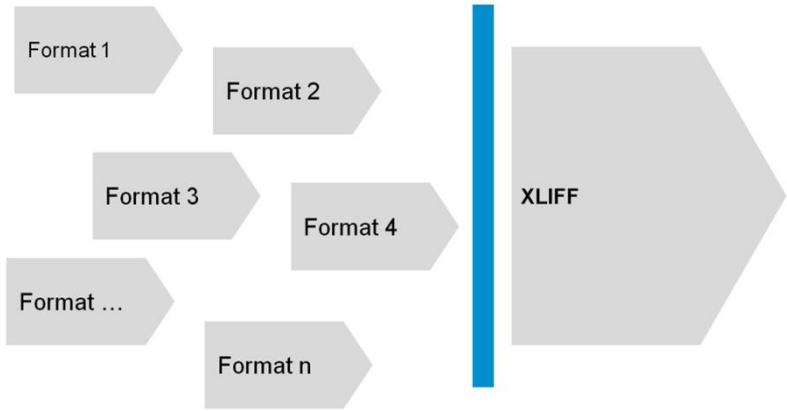
W3C Internationalization Tag Set (ITS)

→ Fight lacking meta data/lacking structure

OASIS XML Localization Interchange File Format (XLIFF)

→ Fight format variability





Bottomline/Hypothesis

You need Natural Language Processing (NLP)/Language Technology (LT) for quality management related to linguistics.

Text Technology is the base for solid, sustainable NLP/LT in real world deployment scenarios.

NLP / LT BASICS

Linguistic/Language Knowledge as Prerequisite

Omnibusse fahrten selten.

What is the language?

How to avoid explicit encoding for *Omnibusses*, *Omnibus?*

How detect spelling mistake?

Lingware – Linguistic/Language Knowledge

Input (“resources/assets”) used by the software

- Statistical model about relative character frequency → input for language identification/detection

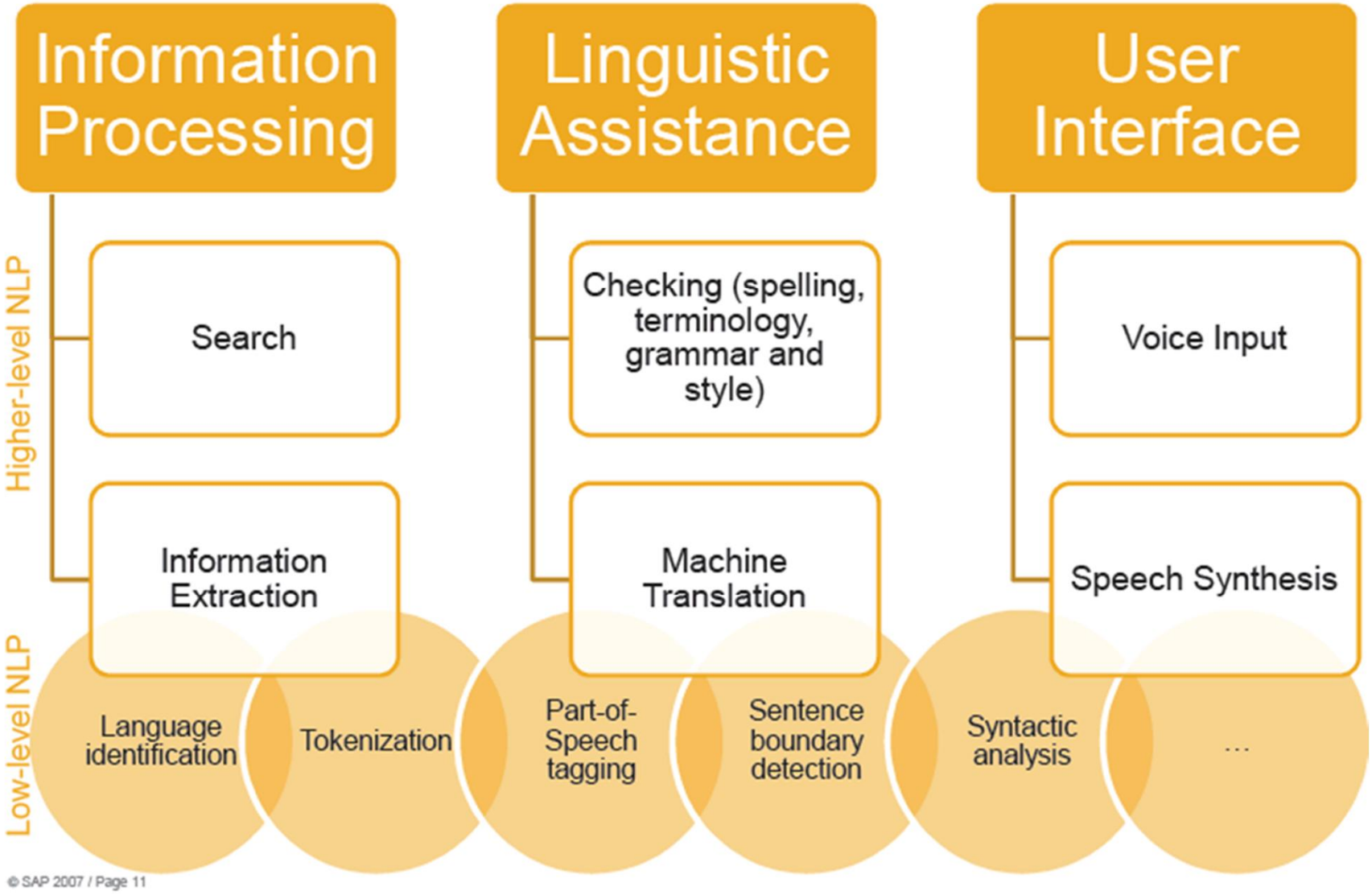
Flavour depends on approach

- Statistical model about translation → statistical Machine Translation (MT)
- Rules for translation → rule-based MT
- Combination of both → hybrid MT

Captive approaches

- Try to reuse your MT lexicon …

Natural Language Processing/Language Technology



Capabilities for Natural Language Processing (1/2)

Text technology provides answers not only related to characters, but also to other areas:

Identify/detect language

- “English”

Tokenize

- #Text#technology#provides# ...
- #テキストテクノロジー#は# ...

Stemming

- “provides” → “provide”

Detect phrases

- #Text technology#

Capabilities for Natural Language Processing (2/2)

Text technology provides answers not only related to characters, but also to other areas:

Check (and correct) terminology, spelling, grammar, style

- Text technlogy provide → Text technology provides

Generate

- Translations, speech from text, text from speech

Match/Lookup

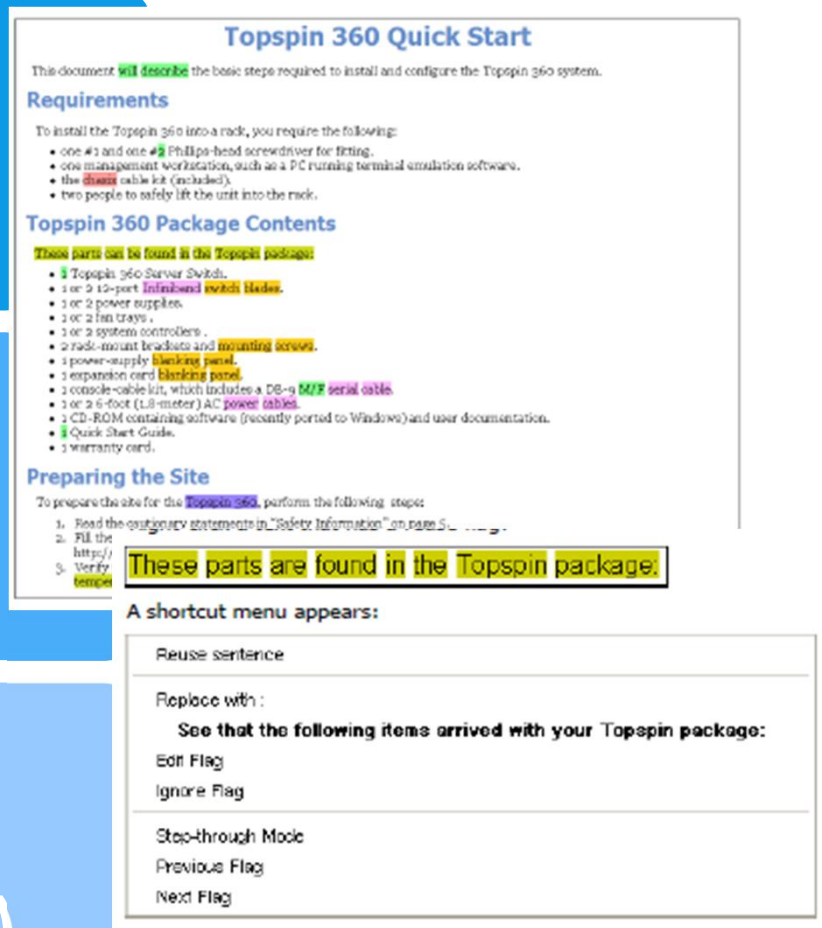
- Against translation memory, term bases, authoring memory

Sample Solution – acrolinx iQ

Check spelling, terminology, grammar, style

Check preference expressions

Harvest/extract terminology (and preference expressions)



TEXT TECHNOLOGY

Foundations for Universal Coverage and Efficiency

Think about content with the world in mind

- Can I encode all characters?
- Can my HTML display the content properly?
- Can I translate efficiently?

Only world-ready NLP/LT is solid and sustainable

Unicode standard

- Allows for content creation and processing in a wide range of languages.
- Applied in many contexts (XML, HTML, multilingual Web addresses like http://ja.wikipedia.org/wiki/東京 , etc.)

Unicode support should be considered as a key feature of any NLP/LT offering.

Dimensions of Text Technology – Overview

- Characters
- Content formats (e.g. HTML, XML, XML-based vocabularies like DocBook or DITA, …)
- Metadata (e.g. Resource Description Framework)
- Filters, e.g. to go from general XML to XLIFF (XML Localization Interchange File Format) based on W3C Internationalization Tag Set (ITS)
- …

Dimensions of Text Technology – Standards

Assets

- Terminology – TermBase eXchange (TBX)
- Former Translations – Translation Memory eXchange (TMX)

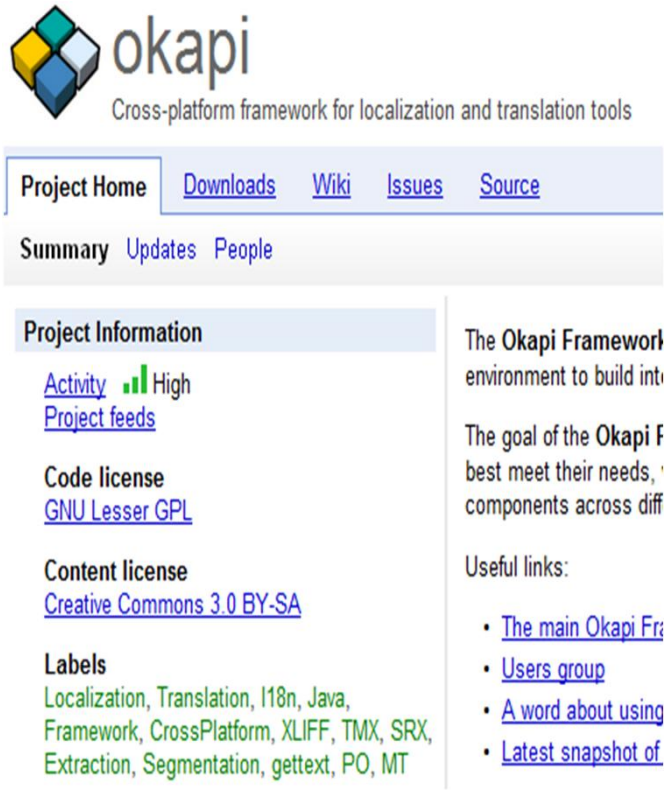
Canonicalized Content

- XML Localization Interchange File Format (XLIFF)

Resource Descriptions

- Internationalization Tag Set (ITS)

okapi - Cross-platform framework for localization and ... - Google Code
code.google.com/p/okapi/
Activity High; Project feeds; Code license; GNU Lesser GPL; Content license ... Okapi artifacts releases: http://repository-okapi.forge.cloudbees.com/release/ ...

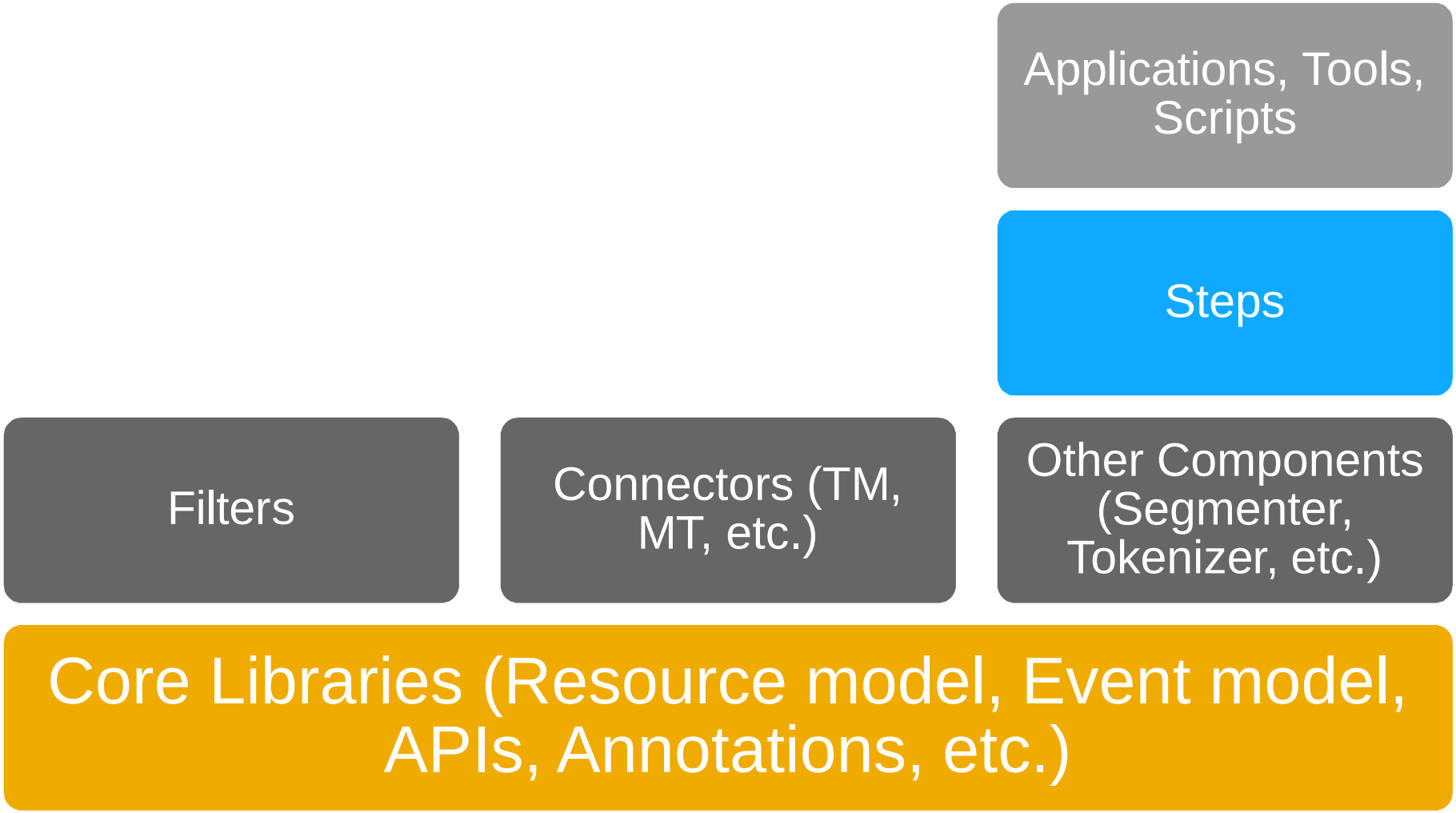


OKAPI / RAINBOW / CHECKMATE

Okapi – Overview

- Open Source set of components and applications to build new or enhance existing content-related processes
- Helps to analyze, fix, convert, package, check quality … chain and couple
- Implements Open Architecture for XML Authoring and Localization Reference Model (OAXAL)
- Roots and strengths in localization and translation
- Uses and promotes open standards where possible
- Various usage options

Okapi – Framework to Build Interoperable Tools



Okapi – Usage Options (Explanation)

- Stand-alone via GUI
- Embedded (e.g. in batch processing server) as Java library
- Coupled as HTTP-accessible server
- From command line

Okapi – Some Out-of-the Box Usage Options (Demo)

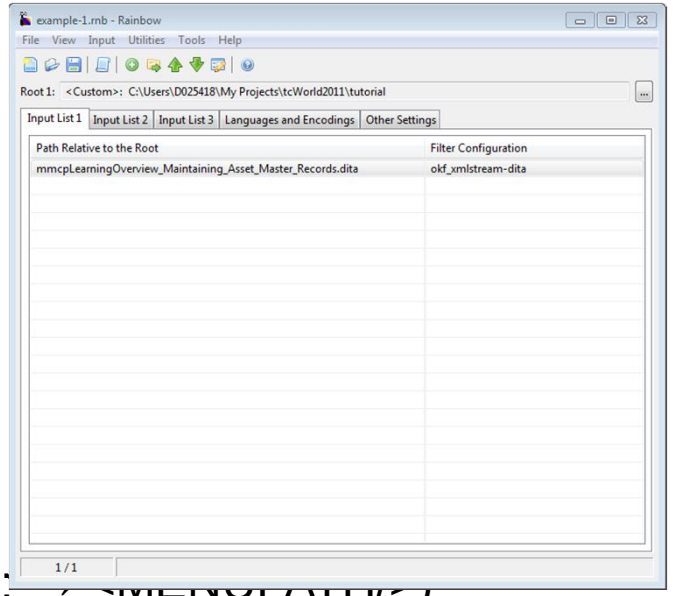
.dita and .xml

UI/Rainbow

Get overview of XML (used characters and markup)

Transform XML

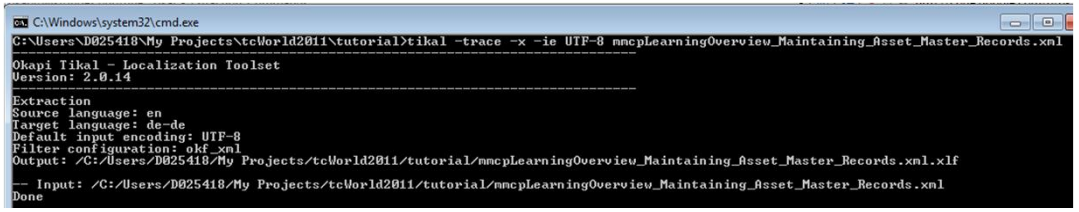
Search and replace (with and without filter; [illegible])



Command Line/Tikal

"okapi-apps_win32-x86_64_0.14\startTikalPrompt.bat"

tikal -trace -x -ie UTF-8 dita.xml



Rainbow / CheckMate – Overview

Rainbow

- Text extraction and merging
- Character encoding conversion
- Term extraction
- File format conversion
- Quality verification
- Translation comparison
- Pseudo-translation
- …

CheckMate

- Quality checks on bilingual translations e.g. on XLIFF and TMX files

Okapi – Advanced Out-of-the Box Usage Options (Demo)

```
<its:rules version="1.0" xmlns:its="http://www.w3.org/2005/11/its">
   <its:translateRule selector="//author" translate="no"/>
   <its:translateRule selector="//title/@content" translate="yes" />
   <its:withinTextRule selector="//i | //b| //u" withinText="yes"/>
</its:rules>
```

Extract term candidates

Generate XLIFF for arbitrary XML

tikal -trace -x -ie UTF-8 its-example.xml

Create ITS rules

tikal -trace -x -fc okf_xml@tcWorld-its -ie UTF-8 its-example.xml

Pseudo-translate XLIFF

Check translated XLIFF

Mismatch in number of inline tags

```
<trans-unit id="2">
   <source xml:lang="en">Open the window <g id="1">later</g>.</source>
   <target xml:lang="de-de">[[[Øpèñ thè wïñðõw .]]]</target>
</trans-unit>
```

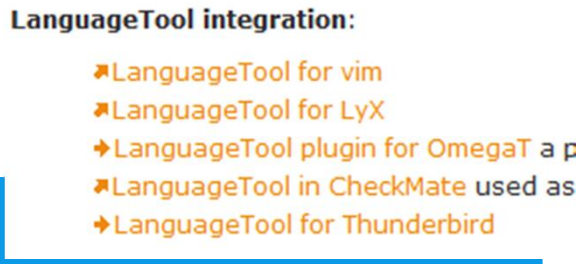
LanguageTool Open Source language checker
www.**languagetool**.org/
Offers open source language and grammar checkers that can be installed as OpenOffice.org extensions.
Demo - Languages - Screenshots - Development

LANGUAGETOOL

LanguageTool – Overview

- Open Source style and grammar checker (no spell checker)
- English, French, German, Polish, Dutch, Romanian, and other languages
- Only knows about errors – not about correct language (rules describe errors)
- Based on NLP/LT (stemming, morphological analysis, part-of-speech tagging)
- Various usage options

LanguageTool – Usage Options (Explanation)



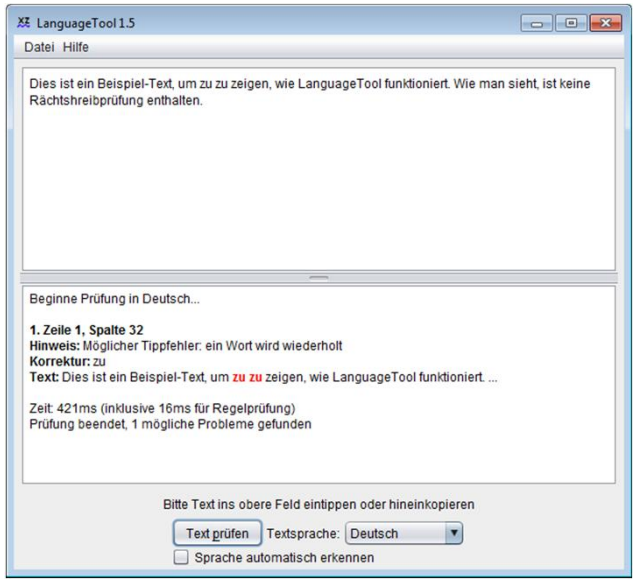
- Embedded (e.g. in OpenOffice/LibreOffice)
- Stand-alone via GUI
- Stand-alone via system tray
- Embedded as Java library
- Coupled as HTTP-accessible service (e.g. from Okapi tools)
- Via output in XML-based format

LanguageTool – Some Out-of-the Box Usage Options (Demo)

Duplicated words …

UI/Stand-alone Editor

```
java -jar LanguageToolGUI.jar
```

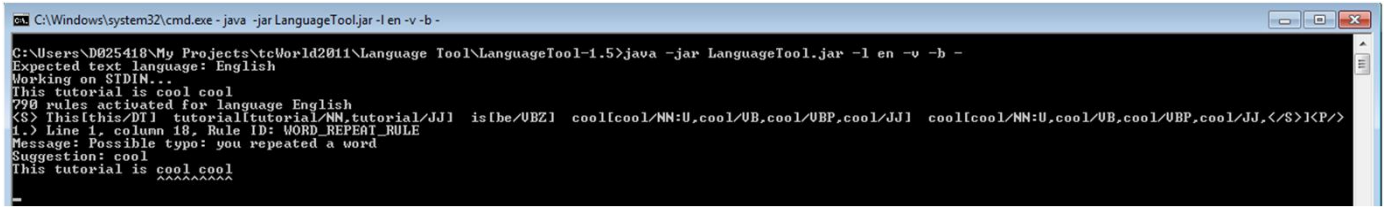


Clipboard “Listener”

```
java -jar LanguageToolGUI.jar –tray
```

Command Line

```
java -jar LanguageTool.jar -l en -v -b -
```



LanguageTool – Checking and Capabilities

- Correct the text directly (rather than only displaying messages and possibly suggestions)
- Special mechanisms for coherency checks, and false friends (mother tongue different from text language)
- Bitext mode (allows checks like length and identity, copy syntax patterns)

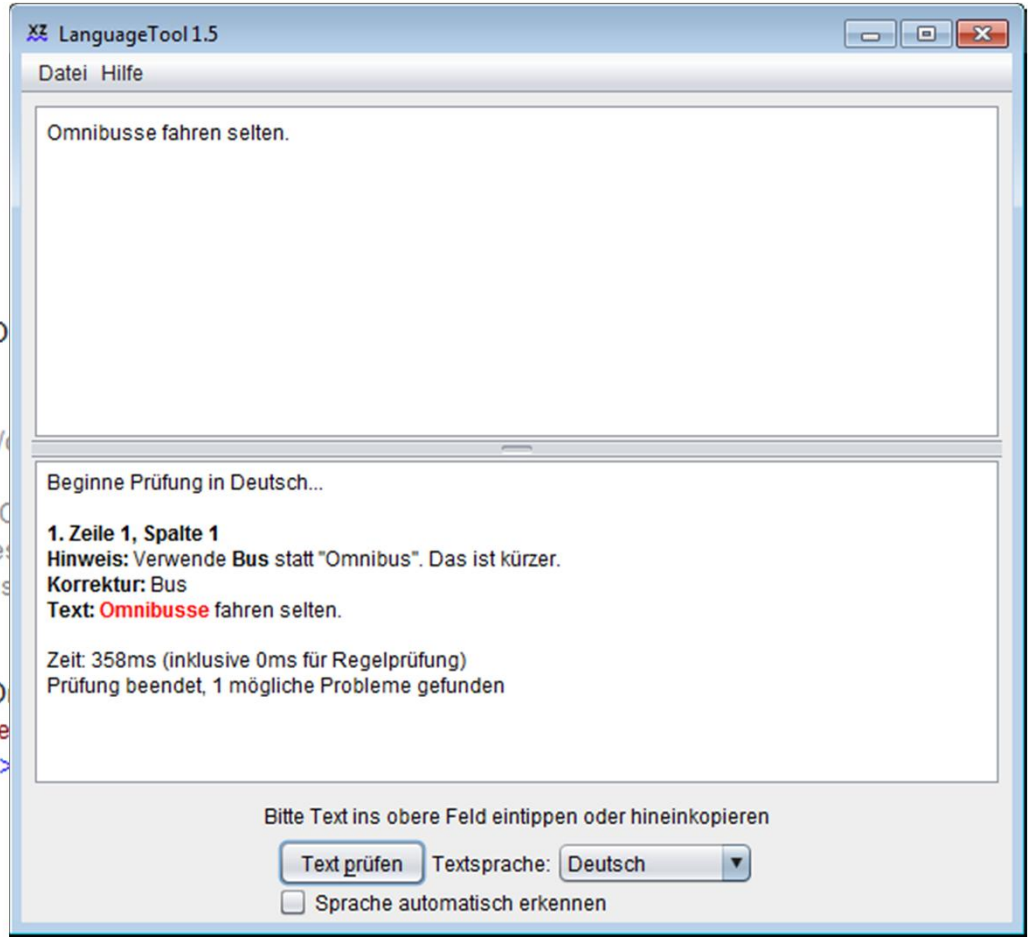
LanguageTool – Customization (Explanation)

- Rules encoded in XML configuration files (easy-to-write, and easy-to-read)
- Rules written in Java (require programming know-how, more powerful)
- Adaptation of existing non-XML resources (e.g. additions to the lexicon)
- Inclusion of processing for additional languages

LanguageTool – Basic Customization (Demo)

rules\de\grammar.xml

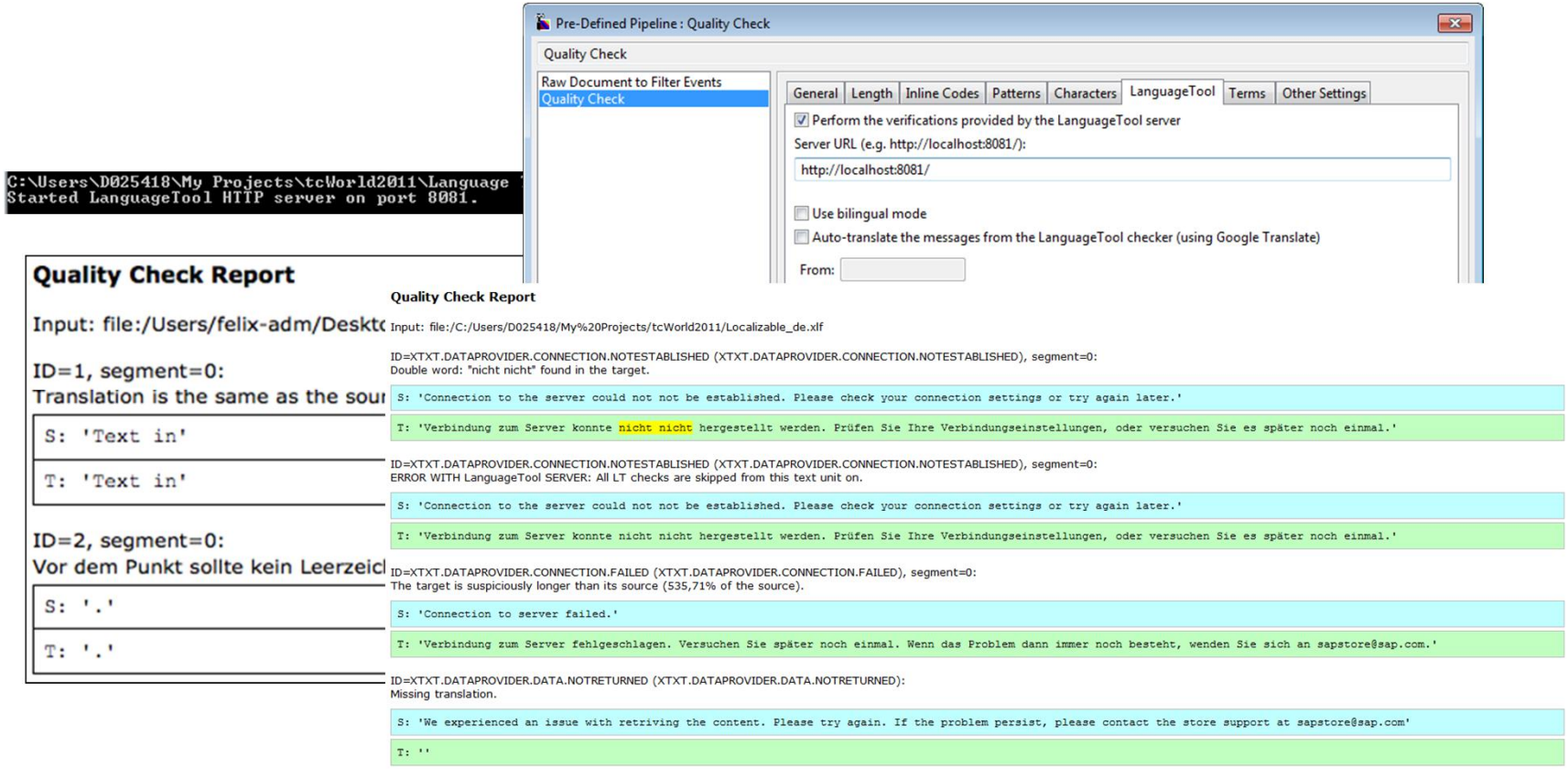
```
<rule id="de_Terminologie_Omnibus-Bus" name="Terminologie: O
    <pattern>
        <!-- CL(Testsatz): Omnibusse fahren selten. -->
        <!-- CL(Referenz): http://www.duden.de/suchen/dudenonline/
        <!-- CL(Ansatz): <token>Omnibus</token> -->
        <!-- CL(Ansatz): <token>Omnibus|Omnibusses|Omnibusse|
        <!-- CL(Ansatz): <token regexp="yes">Omnibus|Omnibusse
        <!-- CL(Ansatz): <token regexp="yes">.+bus|.+busses|.+bus
        <token inflected="yes">Omnibus</token>
    </pattern>
    <message>Verwende <suggestion>Bus</suggestion> statt "O
    <example type="incorrect">Wir fahren günstig mit dem <marke
    <example type="correct">Wir fahren günstig mit dem <marker>
</rule>
```



LanguageTool – Advanced Usage Option (Demo)

Server called via Rainbow / CheckMate

java -cp LanguageTool.jar de.danielnaber.languagetool.server.HTTPServer



LanguageTool – Possible Additional Demos

More grammar/syntax rules related to simple language

java -jar LanguageTool.jar -l de -c UTF-8 -v -b -a tests\tcWorld-Fehler-deDE.txt

java -jar LanguageTool.jar -l en -c UTF-8 --api -b tests\tcWorld-abstract-enUS.txt > tests\output.xml

LUCENE / SOLR

Apache Solr – Overview

Open source full text search server

Built around the Lucene full text indexing system

Can handle input documents in various formats via built-in Tika toolkit

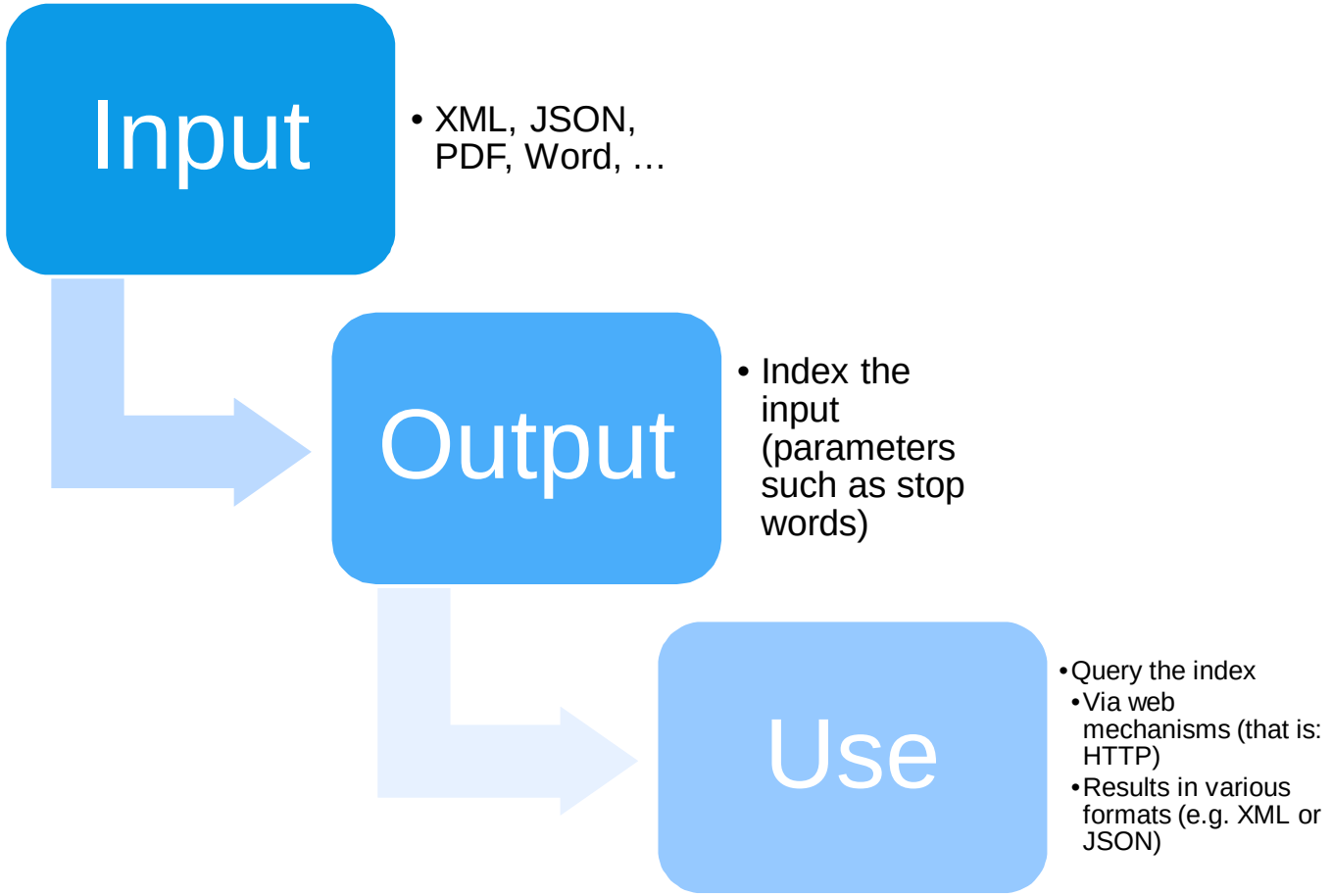
- PDF, XML, doc, docx, …
- Metadata from images, audio, video
- See list of supported formats at http://tika.apache.org/0.10/formats.html

Proven robustness and high performance

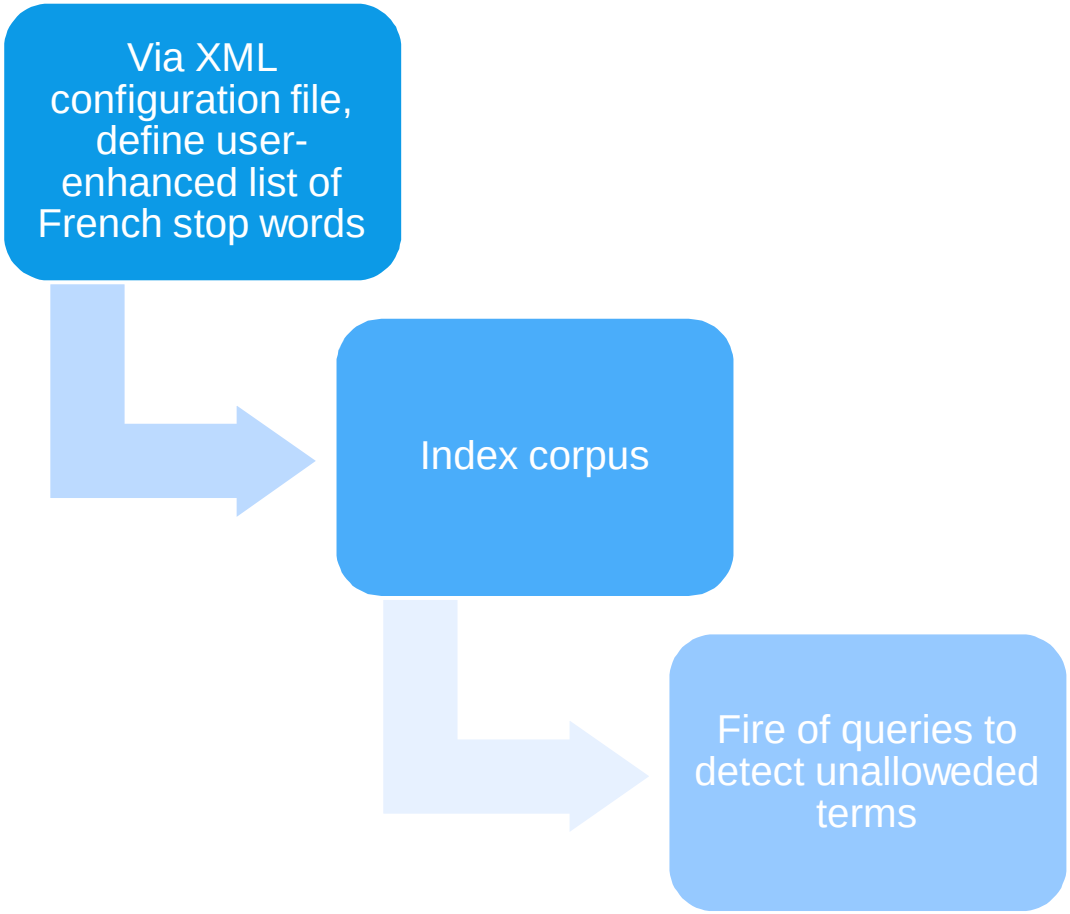
Configuration via XML files – no programming needed

Modules for a large number of languages (see http://wiki.apache.org/solr/LanguageAnalysis for an overview)

Apache Solr – Processing Concept



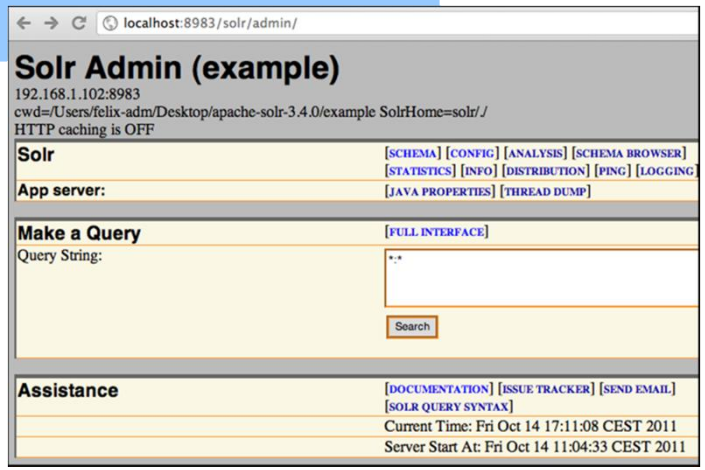
Apache Solr – Sample Quality Assurance Use



Solr – Canned Demo (Set up)

Install Solr
http://lucene.apache.org/solr/index.html#getstarted
(needs Java 1.6)

Open your browser at http://localhost:8983/solr/



Solr – Canned Demo (Add Files via HTTP)

Example URI (e.g. for use with cURL tool)

- http://localhost:8983/solr/update/extract?
  literal.id=doc1&
  uprefix=attr_&
  fmap.content=my_content_de
  &commit=true" -F "myfile=@tutorial.pdf"

Explanation

- Add *tutorial.pdf* to the index
- Use the field *my_content_de* (defined in *schema.xml*)
- ID of the document in the index: *doc1*

Solr – Canned Demo (Add Files via Shell Script)

```
function list_files() {
    if !(test -d $1)
    then echo $1; return;
    fi
    cd $1
    for i in *
    do
        if test -d $i  #if dictionary
        then
            list_files $i #recursively list files
            cd ..
        else
            if [[ "$i" == *html || "$i" == *htm || "$i" == *pdf || "$i" == *doc || "$i" == *docx || "$i" == *png ]]
            then
                count=`expr $count + 1`
                echo "file no $count: $i" #Display File name
                curl "http://localhost:8983/solr/update/extract?literal.id=doc$count&uprefix=attr_&fmap.content=attr_content&commit=true" -F "myfile=@$i"
            fi
        fi
    done
}
java -Ddata=args -jar post.jar "<delete><query>*:*</query></delete>"
if [ $# -eq 0 ]
then list_files .
    exit 0
fi
for i in $*
do
    DIR=$1
    list_files $DIR
    shift 1 #To read next directory/file name
done
```

- Input is a directory name
- Files with specific endings are processed
- Field for indexing is specified in the script – here *attr_content*

Solr – Canned Demo (Query 1)

Get a list of the
50 most frequently
used terms

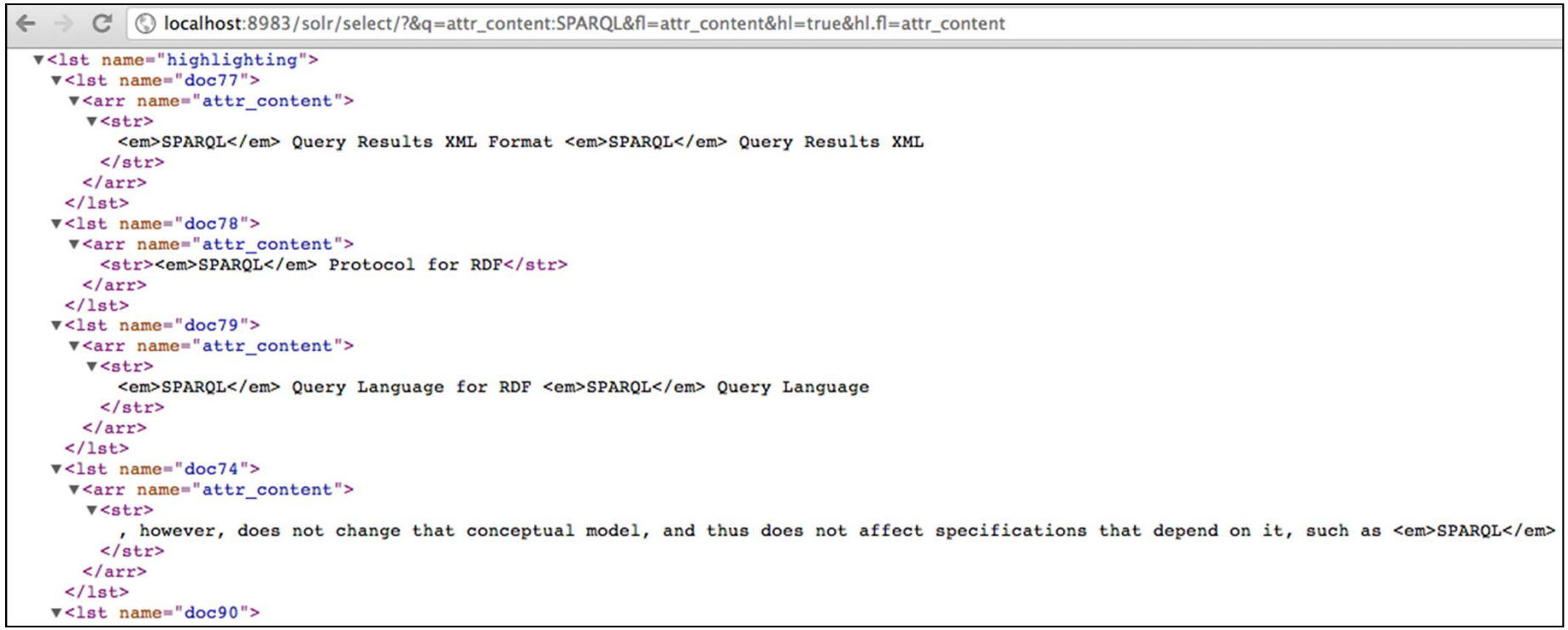
http://localhost:8983/solr/terms?
terms.fl=attr_content&
terms.sort=count&
terms.limit=50

Solr – Canned Demo (Query 2)

Search for a specific term and get documents that contain it

http://localhost:8983/solr/select?&q=attr_content:ArabicNormalizationFilterFactory

Solr – Canned Demo (Highlight Terms in Context)



Solr – Canned Demo (Advanced Use – Configure via schema.xml)

Define “fieldtypes” (and fields making use of fieldtypes) for indexing and query, e.g. language-specific

Example for indexing:

- Use standard tokenizing
- Reference (your) stop words
- Convert to lower case

```
<fieldType name="text_general" class="solr.TextField" positionIncrementGap="100">
  <analyzer type="index">
1   <tokenizer class="solr.StandardTokenizerFactory"/>
2   <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" enablePositionIncrements="true" />
    <!-- in this example, we will only use synonyms at query time
    <filter class="solr.SynonymFilterFactory" synonyms="index_synonyms.txt" ignoreCase="true" expand="false"/>
    -->
3   <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" enablePositionIncrements="true" />
    <filter cla
    <filter cla
  </analyzer>
</fieldType>
```

```
<field name="id" type="string" indexed="true" stored="true" required="true" />
<field name="sku" type="text_en_splitting_tight" indexed="true" stored="true" omitNorms="true"/>
<field name="name" type="text_general" indexed="true" stored="true"/>
<field name="alphaNameSort" type="alphaOnlySort" indexed="true" stored="false"/>
<field name="manu" type="text_general" indexed="true" stored="true" omitNorms="true"/>
<field name="cat" type="string" indexed="true" stored="true" multiValued="true"/>
<field name="features" type="text_general" indexed="true" stored="true" multiValued="true"/>
<field name="includes" type="text_general" indexed="true" stored="true" termVectors="true" termPo
```

Solr – Canned Demo (Language Technology Components)

General components

- http://wiki.apache.org/solr/AnalyzersTokenizersTokenFilters

Language specific components

- http://wiki.apache.org/solr/LanguageAnalysis

```
<fieldType name="text_en" class="solr.TextField" positionIncrementGap="100">
  <analyzer type="index">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory"
            ignoreCase="true"
            words="stopwords_en.txt"
            enablePositionIncrements="true"
            />
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.EnglishPossessiveFilterFactory"/>
    <filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>
    <filter class="solr.PorterStemFilterFactory"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt" ignoreCase="true" expand="true"/>
    <filter class="solr.StopFilterFactory"
            ignoreCase="true"
            words="stopwords_en.txt"
            enablePositionIncrements="true"
            />
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.EnglishPossessiveFilterFactory"/>
    <filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>
    <filter class="solr.PorterStemFilterFactory"/>
  </analyzer>
</fieldType>
```

Solr – Canned Demo (field and fieldtype for German)

```
<fieldType name="text_de" class="solr.TextField" positionIncrementGap="100">
    <analyzer type="index">
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.SnowballPorterFilterFactory" language="German2" />
    </analyzer>
    <analyzer type="query">
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.SnowballPorterFilterFactory" language="German2" />
    </analyzer>
  </fieldType>
…
<field name=”my_content_de" type="text_de" indexed="true" stored="true" multiValued="true"/>
```

Solr – Final Remarks

Provides many basic LT processing modules, including language specific tasks

- Example: Tokenizer for Chinese, Japanese, Korean based on statistical model (n-gram)

Quality is not always good

Pipelines architecture of Solr allows to plug in better, alternative modules

- Example: Better morphological analysis and word segmentation for Chinese, Japanese and Korean

Unfortunately, alternative modules are often commercial and expensive

UIMA

UIMA – Overview

- NLP/LT often needs to analyze unstructured (textual) content in various steps (language identification, tokenization, segmentation, …)
- Need for tying together and orchestrating processing components
- The “Unstructured Information Management Architecture” (UIMA) is an architecture for defining and combining components related to the processing of unstructured information
- Not a piece of running software
- Helps to design processing pipelines (to ease integration efforts and enhance interoperability)
- Recently UIMA became famous via the Watson text analysis system

UIMA – Usage Scenario

Couple Solr with the Alchemy API (allows among others for advanced language identification or named entity extraction)

Details: http://wiki.apache.org/solr/SolrUIMA

CONCLUSION AND OUTLOOK

Conclusions and Outlook

Linguistic quality assurance based on NLP/LT is a reality.

The use of standards and best practices originating in text technology, frameworks for processing pipelines, and open source offerings put it into reach for new constituencies such as small enterprises.

The open source community, as well as stewards of standards and inclusion – such as the World Wide Web Consortium (W3C), and the European Commission (EC) – continue their support for easily accessible multilingual content. Two examples are the EC-funded, and W3C-coordinated Thematic Network “Multilingual Web” (see http://www.multilingual-web.eu) and the forthcoming MultilingualWeb-LT project.

Thank You!

Contact information:

Christian Lieske
christian.lieske@sap.com
www.sap.com

Felix Sasaki
felix.sasaki@dfki.de
www.dfki.de

The authors would like to thank Daniel Naber and Yves Savourel for reviewing draft versions of this presentation.

Disclaimer

All product and service names mentioned and associated logos displayed are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

This document may contain only intended strategies, developments, and is not intended to be binding upon the authors or their employers to any particular course of business, product strategy, and/or development. The authors or their employers assume no responsibility for errors or omissions in this document. The authors or their employers do not warrant the accuracy or completeness of the information, text, graphics, links, or other items contained within this material. This document is provided without a warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

The authors or their employers shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials. This limitation shall not apply in cases of intent or gross negligence.

The authors have no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third-party Web pages nor provide any warranty whatsoever relating to third-party Web pages.

Full Text of the Conference Proceedings

Content Quality Management with Open Source Language Technology

Christian Lieske, SAP AG; Felix Sasaki, DFKI/FH Potsdam

Overview

Today's content production is high paced. In addition, it is demanding in terms of quality and cost efficiency. Furthermore, it is distributed amongst many parties and challenging due to vast volumes. Accordingly, quality management and quality control related especially to textual content is in desperate need of automation. Promising automation solutions related to the linguistic quality of mono- or multilingual textual content rely on Natural Language Processing (NLP)/Language Technology (LT). In order to deploy NLP/LT in real world scenarios, these scenarios need to be seamlessly coupled with, or integrated into content production process chains.

This text – a complement to the corresponding tutorial at the tcWorld 2011 – starts with an overview of typical linguistic quality tasks related to textual content. Afterwards - following a short explanation of general NLP/LT concepts - it sketches NLP/LT for linguistic quality tasks. As a basis for sustainable and solid NLP/LT, so-called Text Technology is being surveyed. The text then moves on to observations related to the use of NLP/LT in real world deployment scenarios.

The concepts are exemplified based on the reality in some technical communications environments with open source offerings. In order to allow for easy follow-up, the demonstrations use open source technology. Some tasks use examples from the realm of language-related requirements related to accessibility (sometimes referred to as easy/simple language).

Linguistic Quality Management and Control for Textual Content

For the purpose of this text, a distinction is made between two different dimensions of quality management and control for textual content (samples are drawn from Web-related content such as HTML pages):

a. Technical - Characters are displayed properly, markup is adequate, links are correct
b. Linguistic - Spelling, grammar, style, and terminology are immaculate

A closer look at the linguistic dimension shows that it encompasses two quite different areas: general information processing on the one hand, and linguistic assistance on the other hand. Information processing often is tied to the initial creation of resources (e.g. databases related to terminology, text chunks, or existing translations) whereas linguistic assistance is mainly encountered during the actual creation of textual content (which of course often relates to, or adapts aforementioned existing resources such as termbases, authoring memories, or translation memories). Example activities and usage scenarios in both areas are the following:

a. Information Processing (search, extract, or enhance; create resources)

- Generate a monolingual list of term candidates
- Align two monolingual texts (e.g. German and French) to fill a translation memory
- Identify phone numbers, addresses etc. in order to tag them with special markup

b. Linguistic Assistance (propose and check; help to author or translate efficiently and properly)

- Suggest – possibly alternative – wording (based for example on entries in termbases, or authoring/translation memories)
- Check – and possibly automatically correct – texts against rules for spelling, terminology, grammar, or specific style rules (e.g. company-proprietary guidelines)
- Transfer/translate text – either by reusing and possibly adapted (sub-segment) strings, or generating via Machine Translation – from a source language to a target language

In the realm of technical communications, very often four roles come into play in the aforementioned activities:

- Authors want interactive assistance for creating of correcting texts
- Managers like to see numbers (e.g. quality indicators based on the number of errors calculated by a tool) and want to use them in analytical applications
- Solution architects, terminologists and translators create resources (e.g. termbases and formalized style rules), technical components, and processing pipelines
- Translators want to match against termbases and translation memories

Basics of Natural Language Processing (NLP)/Language Technology (LT)

Automation related to the linguistic tasks mentioned above usually needs to go beyond simple strings processing approaches like the use of regular expressions. In one shape or other, linguistic knowledge - a model of the language at hand - is needed. This is the focus of Natural Language Processing (NLP)/Language Technology (LT). To be specific, NLP/LT most often comprises two main – very often intertangled – ingredients (the details provided are just examples):

a. Software, that is components for

- Identifying the language in which a text is written
- Applying tokenization rules (which break strings up into entities such as words)
- Stemming or analyzing morphologically (which determine base forms, or sub-entities of a compound expression)
- Applying segmentation rules (which break documents up into entities such as sentences)
- Computing relationships between entities (e.g. one that groups words into entities such as nominal phrases "Radio Frequency")

- Comparing a given spelling to a model that encodes the correct spelling
- Correcting an incorrect grammatical structure according to a model that encodes the correct grammar
- Generating translations
- Covering special communication modalities (e.g. spontaneous speech)
- Matching against resources such as termbases, authoring memories or translation memories

b. Lingware, that is formalized linguistic knowledge such as

- A statistical model that encodes the relatively frequency of a certain characters in a language (or script); can be used for language identification
- A set of rules that model multiword phrases; can be used to generate term candidates
- A set of mathematical properties (e.g. bi- or trigrams of part-of-speech tags) that capture syntactic features; can be used to detect grammar errors
- A lexicon that captures so-called lemmatas and features (such as their inflection paradigm); can be used for machine translation

As indicated above, lingware pertains to different entities (e.g. characters, words/tokens/ideographs, syntactic structures, ...) and comes in two categories: statistical or symbolic. In case the two categories are combined, the corresponding offerings often are termed "hybrid" (e.g. "hybrid Machine Translation").

Very often you will see a separation of lower level NLP/LT (e.g. tokenizer, part-of-speech tagger, ...) and higher level (term candidate extractor, spell checker, grammar checker, ...). The higher level NLP/LT is what ordinary users get to see.

In today's world, NLP/LT most often starts with an analysis of text. Only after the analysis, possibly a generation of text (e.g. a correction suggestion) is triggered. A simple approach to generation works with templates (where no syntactic structures are generated, but only gaps are filled in a fill-in-the-blank fashion).

Tools that are categorized as "computer assisted authoring/translation" often include approaches that originate in NLP/LT. At least initially, the focus of these tools, however, relates to the creation and reuse of special purpose assets (such as the term candidates, bilingual translated segments, or aligned translated documents of an individual translator or a whole company).

Text Technology

"Text technology" is the foundation for solid and sustainable NLP/LT applications. An example for text technology is the Unicode standard. It is being applied in many contexts (XML, HTML, multilingual Web addresses like http://ja.wikipedia.org/wiki/東京, etc.) and allows for content creation and processing in a wide range of languages. Thus, Unicode support should be considered as a key feature of any NLP/LT offering. In a sense, text technology embodies rules (e.g. for the syntax of a markup language), and best practices related to many sub-domains (e.g. character encoding) within NLP/LT.

Text technology provides answers not only related to characters, but also to other areas:

- Content (e.g. HTML, XML, XML-based vocabularies like DocBook or DITA, ...)
- Metadata (e.g. Resource Description Framework)
- Filters, e.g. to go from general XML to XLIFF (XML Localization Interchange File Format) based on W3C Internationalization Tag Set (ITS)
- ...

Although text technology thus is vital for real-world deployment of NLP/LT, it is often neglected. Example: Instead of reusing an existing vocabulary, simply a new, proprietary one is created. Luckily, however, more and more projects and offerings are dedicated to the integration of text technology with NLP/LT. One example is the m4loc project http://code.google.com/p/m4loc/ . It aims at integrating the statistical machine translation engine Moses with XLIFF based localization workflows.

NLP/LT and Text Technology in real world deployment scenarios

A typical task related to linguistic quality control usually requires that several capabilities/tools/components are seamlessly integrated into each other. A content production process may for example require the following sequence of activities (all of which relate to primary goals such as superb user experience, and legal compliance):

a. Find a special purpose format object such as an image represented as Scalable Vector Graphics (SVG) in a Content Management System (CMS)
b. Convert the SVG file into more "linguistic friendly" XML (e.g. XML Localization Interchange File Format; XLIFF)
c. Check the content of the XLIFF file for unallowed terms (either source language, target language or both; based on a termbase that lists unallowed and allowed terms)
d. Submit the checking result into a Business Intelligence solution (e.g. for reporting purposes)

Accordingly, real world deployment scenarios need to answer questions related to integration approaches, and thus need to be concerned with areas such as the following:

a. Formats (proprietary, Java Script Object Notation, XML, XLIFF, ...)
b. Coupling (Object Linking and Embedding, HTTP, Web Services, ...)
c. Libraries/Application Programming Interfaces/Software Development Kits
d. Orchestration (e.g. synchronization of calls, and "bus-like" integration or annotation framework)

NLP/LT and Text Technology in the Open Source world

Remarks on Basic Due Diligence related to Open Source

Before working with open source offerings, questions such as the following should be addressed:

a. Is this project still alive? – Look at last version, reported bugs, fixes etc.

b. What am I allowed to do? – Look at license/license conditions
c. Is this a solid/quality endeavor? – Look at mailing lists etc.
d. In which languages are the User Interface, and User Assistance/Documentation provided?
e. What is the quality of the documentation? Do additional books exist?
f. Which interfaces/connectors/exchange formats are supported?
g. Is the technology behind mainstream (or is for example the programming language one that is almost unknown)?
h. What might be the unique selling proposition of commercial offerings? – Look for details such as supported languages, accuracy, quotes from reference customers

Okapi/Rainbow/CheckMate

Linguistic Quality Management can only be successful if it is seamlessly integrated into an overall content production chain. Accordingly, there is very often a need to "massage" content (e.g. to convert it), or to have content flow from one processing component to another one.

"The Okapi Framework is a cross-platform and free open-source set of components and applications allows process architects to build new content-related processes or enhance existing ones while preserving compatibility and interoperability. The framework has its roots and strengths in processes and tasks related to localization and translation. Whenever possible, the project uses and promotes open standards.

Rainbow is an Okapi-based GUI application to launch various utilities such as: Text extraction (to XLIFF, OmegaT projects, RTF, etc.) and merging, encoding conversion, terms extraction, file format conversions, quality verification, translation comparison, search and replace on filtered text, pseudo-translation, and much more. Using the framework's pipeline mechanism, you can use Rainbow to create chains of steps that perform specific set of tasks specific to your needs. CheckMate is an Okapi-based GUI application that performs various quality checks on bilingual translation files such as XLIFF, TMX." (adapted from http://okapi.opentag.com/).

Usage scenarios covered during the tutorial:

- Used characters listing
- Term extraction
- Conversion from proprietary XML to XLIFF based on W3C ITS
- ...

LanguageTool

LanguageTool is an Open Source style and grammar checker for English, French, German, Polish, Dutch, Romanian, and other languages. It does not include spell checking. LanguageTool, however, can be used for tasks related to terminology control (adapted from http://www.languagetool.org/).

LanguageTool is based on NLP/LT concepts such as stemming and morphological analysis, and part-of-speech (POS) tagging. It can be used out-of-the-box. In addition,

however, it is a framework for building language processing – in particular language quality assurance – applications.

LanguageTool only knows about errors – not about correct language. All rules describe errors.

Important features of LanguageTool are the following:

1. Various customization options

    - Rules encoded in XML configuration files (easy-to-write, and easy-to-read)
    - Rules written in Java (require programming know-how, more powerful)
    - Adaptation of existing non-XML resources (e.g. additions to the lexicon)
    - Inclusion of processing for additional languages

2. Solid foundation for use and connectivity

    - Embedded (e.g. in OpenOffice/LibreOffice)
    - Stand-alone via GUI
    - Stand-alone via system tray
    - Embedded as Java library
    - Coupled as HTTP-accessible service (e.g. from Okapi CheckMate)
    - Via output in XML-based format

3. Checking and correction capabilities

    - Correct the text directly rather than only displaying messages and possibly suggestions
    - Special mechanisms for coherency checks, and false friends (mother tongue different from text language)
    - Bitext mode that allows checks like length and identity, copy syntax patterns

Usage scenarios covered during the tutorial:

- Out-of-the box rules with various interfaces (e.g. UI)
- Various grammar/syntax rules related to simple language
- Term checking (possibly bilingual)

Tips&Tricks that might be mentioned:

- Inflection/deinflection only possible if known (to Morphy for German)
- Names only used in configuration/rule selection dialogue
- XML entities can be used to ensure consistent messages
- Example can be used for unit testing (testrules.bat)
- Additional rules can even be loaded via WebStart
- Coherence rules (implemented in Java) help to ensure proper terminology

Advanced Topics that might be touched upon:

- Rule for - (use of decomposition)

Apache Solr/Apache Lucene

Apache Solr is an open source full text search server, built around the Lucene full text indexing system. Solr's processing concept is simple:

a. You put documents in various formats (XML, JSON, PDF, Word, ...) in; extraction is performed via http://tika.apache.org/. Tika plays an important role in a couple of extraction and roundtripping scenarios (e.g. in the Moses4Localization project).
b. Solr creates an index for the documents; you can fine tune the indexing via parameters such as stop words
c. You query the documents/search the index via the Web mechanisms (that is: HTTP) and receive the result in various formats (e.g. XML or JSON)

Besides its robustness and high performance, key feature of Solr relevant for "linguistic quality assurance" are the following:

- Configuration based on XML file – no programming is needed.
  Example: In an XML configuration file, a field is defined that can be used for indexing of French text. This definition related to a user-enhanced list of stop words

- Modules for a large number of languages (see http://wiki.apache.org/solr/LanguageAnalysis for an overview) – allows for fulfilling the "linguistic quality assurance" task in a multilingual fashion (e.g. via checks against lists of unallowed terms)

An example technical communications usage scenario for Solr:

> You may have to assure that only allowed terms are used in a large body of text. Solr indexes your texts. Once Solr has created the index, you do your term checks against Solr's index. This approach may reduce the time that is needed for your checks down from several hours to several minutes.

Unstructured Information Management Architecture

NLP/LT often needs to analyze unstructured (textual) content in various steps (language identification, tokenization, segmentation, ...). Thus, there is a need for tying together and orchestrating processing components.

The "Unstructured Information Management Architecture" (UIMA) is a framework for defining and combining components related to the processing of unstructured information. Recently UIMA became famous via the Watson text analysis system – the system (which is based on UIMA) won a Jeopardy competition against several "human" Jeopardy stars.

Unlike Okapi/Rainbow/CheckMate, Language Tool or Solr, UIMA is not a framework to use "as is". The UIMA framework itself provides no ready to use filters, language

checkers, indexing components, or the like. UIMA rather helps to design processing pipelines. Furthermore, UIMA eases integration efforts and enhances interoperability since it outlines interfaces between components/pipelines steps. Example: Using UIMA (see http://wiki.apache.org/solr/SolrUIMA) Solr can be easily coupled e.g. with the Alchemy API, which allows among others for advanced language identification or named entity extraction.

Conclusion and Outlook

Linguistic quality assurance based on NLP/LT is a reality. The use of standards and best practices originating in text technology, frameworks for processing pipelines, and open source offerings put it into reach for new constituencies such as small enterprises.

The open source community, as well as the stewards of standards and inclusion – such as the World Wide Web Consortium (W3C) and the European Commission (EC) – continue their support for easily accessible multilingual content. Two examples are the EC-funded, and W3C-coordinated Thematic Network "Multilingual Web" (see http://www.multilingual-web.eu) and the forthcoming MultilingualWeb-LT project.

Appendix

Intended Demo Blocks

- Linguistic checks from withing LibreOffice/OpenOffice
- Term candidate generation (using frequencies) based on Solr/Lucene
- Adaptation of resources for LanguageTool
- Format conversions with Okapi/Rainbow
- Translation-related checks with CheckMate and LanguageTool

Authors' Addresses

Christian Lieske, SAP AG, christian.lieske@sap.com
Felix Sasaki, DFKI/FH Potsdam, felix.sasaki@dfki.de

References

References can be found in the body of the text. The authors welcome any questions related to possible additional references.

Disclaimer

the accuracy or completeness of the information, text, graphics, links, or other items contained within this material. This document is provided without a warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. The authors or their employers shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials. This limitation shall not apply in cases of intent or gross negligence. The authors have no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third-party Web pages nor provide any warranty whatsoever relating to third-party Web pages.