Regeln erstellen mit dem Language Tool

Einfaches Vorgehen

Wenn du neue Regeln erstellen möchtest, solltest du diese ausgiebig testen. Es gibt unterschiedliche Möglichkeiten Regeln zu testen und zu erstellen. Dies wird im Folgenden genauer erläutert.

Regel Editor

Mit dem Regel-Editor auf der Seite des Language Tools, können einfache Regeln erstellt oder mit „Parese existing XML“ bereits bestehende Regeln geprüft werden. Dort werden die einzelnen Bestandteile analysiert. Im „Expert Mode“ können komplexere Regeln geschrieben werden. Es werden Wikipedia Artikel als Testkorpus verwendet, um die neu erstellte Regel zu testen. Der Sinn dahinter besteht zuallererst darin, die Regel zu testen und falls die Regel falschen Alarm auslöst, dies zu analysieren. Wenn die Regel mit dem Textkorpus fehlerfrei funktioniert, sollte die Regel noch genauer verfasst werden, um falsche Fehlermeldungen in Zukunft zu vermeiden.

Den Regel Editor findest du unter: https://community.languagetool.org/ruleEditor2/index?lang=de

Local Plain Text Checks

Wenn du ein Entwickler-Setup (Java und Maven) besitzt, kannst du mit *languagetool-standalone/scripts/regression-test.sh.* arbeiten. Gehe in das Verzeichnis, in dem sich das Skript befindet, und rufe dieses auf. Danach rufe das Skript mit den erforderlichen Parametern auf und nehme Änderungen an den LanguageTool-Relgen und /oder dem Quellcode vor. Daraufhin führst du das Skript noch einmal mit dem gleichen Parameter aus. Dies hat zur Folge, dass ein Diff-Tag des alten gegenüber dem neuen Ergebnis gedruckt wird und dadurch alle neu entstandenen Fehlalarme erkennbar werden. Um dies zu ermöglichen, benötigt es eine große plain-text Datei, die im LangaugeTool ausgeführt werden kann.

Lokale Wikipedia Prüfung

Um die Regeln mit einer größeren Menge Wikipedia-Dokumenten zu überprüfen, muss der LanguageTool-Wikipedia Snapshot heruntergeladen werden. Dafür müssen die Dateien entpackt und danach ausgeführt werden.

Den LanguageTool-Wikipedia Snapshot findest du unter: http://www.languagetool.org/download/snapshots/?C=M;O=D

java -jar languagetool-wikipedia.jar

Der oben gezeigte Ausschnitt zeigt die Verwendung an. Diese Verwendung wird dann mit check-data ergänzt:

java -jar languagetool-wikipedia.jar check-data

Nachdem du diese Schritte durchgeführt hast, lade einen Dump von http://dumps.wikimedia.org/dewiki/latest/ herunter. Bei dem Link handelt es sich um den deutschen Link. Wenn du dich für eine andere Sprache interessierst, tausche das „de“ in der URL mit dem Code der Sprache aus. Die verwendete Datei wird für die Verwendung *„dewiki-neuste-page-articles.xml.bz2“* empfohlen. Auch dort kann der gewünschte Sprachcode anstelle von „de“ verwendet werden. Danach entpackst du den Speicher. Um eine Regel mit ID MY_RULE zu überprüfen, müssen ähnliche Befehle verwendet werden:

java -jar languagetool-wikipedia.jar check-data -l de -f dewiki-20141006-pages-articles1.xml -r MY_RULE --max-errors 100

Denke immer daran de zu ersetzten, wenn du einen anderen Sprachcode verwenden möchtest. Der Befehl überprüft die Artikel aus dem Dump (Speicherauszug) gegen die Regel MY_RULE und stoppt, wenn 100 Übereinstimmungen vorliegen.

Anmerkungen:

- Manche Wikipedia-Artikel sind sehr komplex und für diese ist es meist nicht möglich, die Regeln gegen den gesamten Speicherauszug zu prüfen. Durch das Definition eines Limits kann die maximale Anzahl der zu prüfenden Artikeln festgelegt werden.
- Da die Textextraktion von Wikipedia noch nicht sehr robust ist, ist eine Verwendung dieses Ansatzes für Regeln, die eine Verwendung von Leerzeichen oder Sonderzeichen überprüfen, nicht ratsam.
- Die Option -f kann mehrmals angegeben werden. Um den enzyklopädischen Stil von Wikipedia zu ergänzen, muss die herunterladbare Datei von Tatoeba angeben werden: *http://downloads.tatoeba.org/exports/sentences.tar.bz2*
  Du musst die Datei entpacken und die Zeilen herausfiltern, die die Sprache enthalten, an der du interessiert bist. Dies geschieht durch: z. mit grep -P "\ teng \ t" sets., um die englischen Sätze zu erhalten.

Komplexes Vorgehen

Die vorherigen Tests sind sehr einfach und nützlich, aber decken nur die Prüfung mit Wikipedia-Texten ab. Es ist von Vorteil, die erstellten Regeln gegen einen vielschichtigen Satz von Texten zu testen.

Arbeit mit einer sinnvollen Korpussammlung

Um eine eigene Korpussammlung zu erstellen, suchst du beispielsweise nach mehreren Texten aus unterschiedlichen Disziplinen oder Genres. Außerdem können

auch Artikel oder andere kurze Textelemente gesammelt werden. Dadurch können schnell Level 1 Test durchgeführt werden. (In unserem Fall haben wir einen Textkorpus aus Texten mit normaler Sprache und Leichter Sprache erstellt, um eine Gegenüberstellung zu ermöglichen.)

Verwendung von separaten Regeldateien

Deine Sprachregeldatei befindet sich unter: *org/groundtool/rules/xx/grammar.xml.* Dabei stellt *xx* einen Sprachcode wie *en* oder *de* dar. Dabei handelt es sich um die Standard-Regeldatei und es wird die Datei LanguageTool geladen, wenn du diese ausführst. Es ist von Vorteil, eine separate XML-Datei zu erstellen, die die Regeln der Entwicklung enthält. Dadurch erhältst du einige Vorteile:

- Du arbeitest mit einem kleinen Satz von Regeln, egal wie groß die Sprache grammar.xml-Datei ist.
- Es lassen sich die Leistungen verschiedener Ansätze für das gleiche Produkt vergleichen.
- Die Grammatikprüfung wird schneller durchgeführt.

Die Datei wird jedoch nur von dem eigenständigen GUI (languagetool.jar) unterstützt. Bei Bedarf kannst du die Arbeitsdatei *rules-xx-Actual.xml* herunterladen und deine Eingaben auf kurze Dateien testen.

Erstellen von Beispieldateien

In den LanguageTool-Textdatein lassen sich Befehle eingeben, daher solltest du einige Dateien mit Beispielsätzen einrichten. Für jede Regel oder Regelgruppe sollte eine Datei „sollte warnen“ und eine Datei „sollte nicht warnen“ erstellt werden. Jede Datei enthält nur einen Satz pro Zeile.

Wenn eine einfache Regel vorliegt, müssen nicht immer zwei Dateien erstellt werden. Es muss sichergestellt werden, dass die richtigen Beispiele am Ende der Datei stehen.

Entwickler Modell

1. Wähle die Regel, die du implementieren oder erstellen möchtest
    - Zuerst solltest du nach der Regel suchen, die du erstellen möchtest. Es kann sein, dass diese schon implementiert wurde oder in einer anderen Regel enthalten ist.
    - Wenn dies nicht der Fall ist, solltest du sichergehen, dass die Regel kompatibel ist. Es gibt einige Regeln, die auf Bedeutungen beruhen, wie mehrdeutige Konstruktionen.
    - Vermeide Regeln, die nicht sehr häufig auftreten. Sie verschwenden Computerressourcen und helfen nur wenigen Nutzern.

2. Entwerfe die Regel mithilfe einer Regel oder Regelgruppe

- Wenn eine Regel zu komplex ist, erstelle eine separate XML Datei, die die neue XML Regel beinhaltet. Wenn du dies umsetzten möchtest, musst du die grammar.xml Datei als Template verwenden.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="../print.xsl"  title="Pretty print"
?>
<?xml-stylesheet type="text/css" href="../rules.css"  title="Easy editing
stylesheet" ?>
<rules        lang="es"        xsi:noNamespaceSchemaLocation="../rules.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- Here goes the stuff -->
</rules>
```

Aktualisiere den Sprachcode. Verwende beispielsweise xx als Sprachcode. Es besteht außerdem die Möglichkeit, die Datei zu verändern. Dazu müssen die Stylesheet Paths aktualisiert werden.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="rules/print.xsl" title="Pretty print"
?>
<?xml-stylesheet type="text/css" href="rules/rules.css" title="Easy editing
stylesheet" ?>
<rules        lang="xx"        xsi:noNamespaceSchemaLocation="../rules.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- here goes the stuff -->
</rules>
```

Danach muss die Datei gespeichert werden. Um es mit *languagetool.jar* zu verwenden, sollte ein Name wie *rules-xx-Actual.xml* erstellt werden. Lade dann die Regel-Datei des Datei-Menüs herunter. Danach kannst du mit dem kleinen Regelset arbeiten.

Suche nach Anwendungsfällen und schreibe diese in Dateien

- Suche nach unterschiedlichen Fällen, die der Regel entsprechen und bei denen eine Warnung ausgegeben werden soll. Schreibe diese in die Datei: „soll warnen“.
- Suche nach Fällen, die die Regel täuschen könnten. Diese sollen in die Datei „sollte nicht warnen“ geschrieben werden oder am Ende der eindeutigen Datei.

Es ist immer eine gute Idee, „Greedy-Ruels“ zu schreiben und damit den Korpus zu testen und die Ergebnisse zu sichern. In späteren Schritten kann dann die Feinabstimmungsregel mit allen Fällen am Korpus überprüft werden.

3. Teste die neue Regel mit languagetool.jar.
    - Mit languagetool.jar kann sehr schnell getestet werden, ob die letzte Modifizierung funktionstüchtig ist. Vergiss nicht deine Entwicklungsdatei zu wählen! Bearbeite diese so lange, bis du damit zufrieden bist.

4. Teste dann die Regel gegen ihre „soll warnen“- Dateien und „soll nicht warnen“-Dateien
    - Die Ausgabe sollte für zukünftige Referenzen behalten werden. Es besteht die Möglichkeit einige Fälle zu beheben, jedoch können dann andere aufgehoben werden, wenn die Regel geändert wird. Wenn du unzufrieden mit deiner Regel bist, kehr einfach zu Schritt 2 zurück.

5. Integriere/ ersetzte die neuen Regeln in die reguläre Datei *grammara.xml* und führe tesrules.sh (Linux) testrules.bat (Windows) aus
    - Dieser Schritt ist essenziell, da dadurch viele Probleme in den neuen Regeln anhand von Beispielen gefunden werden können. Es können auch Fehler, wie nicht markierte reguläre Ausdrücke erkannt werden. Nachdem *testrules.sh* oder *testrules.bat* ausgeführt wurde, können die neuen Regeln an deiner Korpussammlung getestet werden. Das Skript muss dann mit dem Sprachcode als Parameter aufgerufen werden, um die Tests innerhalb der gewählten Sprache auszuführen.

6. Teste jetzt deine Regeln gegen deine Korpussammlung der 1, Ebene
    - Dabei handelt es sich um das gleiche Verfahren wie in Schritt 4 aber unter der Verwendung der kurzen Artikel. Mit einem einfachen Shell-Skript kann dieser Vorgang automatisiert werden. Das folgende Beispiel basiert darauf, dass alle Texte in einem Verzeichnis mit dem Namen texts enthalten sind und das gleiche Präfix lt- (für langen Text) benutzt wird:

```
#!/bin/bash
# testes-tl.sh
# Long test for Spanish grammar
# Copyright (C) 2010 Juan Martorell

_help()
{
  echo "$0 {level}"
  echo
  echo '  where {level} is one of'
  echo '       1: Test level 1 corpus: files prefixed with lt-1-'
  echo '       2: Test level 2 corpus: files prefixed with lt-2-'
  echo '       a: Test all levels'
  exit
}

function process {
for file in $FILELIST
```

```
    {
        echo "Processing $file"
        java -jar languagetool-commandline.jar --language es \
        --disable
WHITESPACE_RULE,UNPAIRED_BRACKETS,COMMA_PARENTHESIS_WHITESPACE,DOUBLE_PUNCT
UATION \
         $file >$file.log
    }
}

if [ -z "$1" ]; then
   _help
elif [ "$1" == "1" ]; then
  FILELIST=`ls -rS texts/tl-1-*.txt`
  process
elif [ "$1" == "2" ]; then
  FILELIST=`ls -rS texts/tl-2-*.txt`
  process
elif [ "$1" == "a" ]; then
  FILELIST=`ls -rS texts/tl-?-*.txt`
  process
elif [ "$1" == "--help" ]; then
   _help
else
  echo "** $1 is not a valid option."
  echo
   _help
fi
exit
```

7. Teste deine Regeln gegen deine Korpussammlung Level 2
    - Wenn du mit deinem Testergebnis der Stufe 1 zufrieden bist, müssen mindestens 1 Mio. Wörter getestet werden. Dies kann viel Zeit beanspruchen, aber danach sollten alle Fehlalarme verbessert sein.

8. Teste dann die reguläre Regeldatei gegen deine Korpussammlung der 2. Ebene
    - Kopiere die Regel wie in Schritt 5 beschrieben in die reguläre Datei *grammar.xml* und führe die Tests mit dem Korpus der Ebene 2 durch. Beispielsweise kann folgendes Skript dafür verwendet werden:

```
#!/bin/bash
# testes-tl-all.sh
# Complete test for Spanish grammar
# Copyright (C) 2010 Juan Martorell

function scancorpus {
for file in `ls -rS texts/tl-*.txt`
    {
        echo "Processing $file"
        java -jar languagetool-commandline.jar --language es --mothertongue
en $file >$file.log
    }
}
scancorpus
```

Hier kann sehr gut nach überlappenden Regeln gesucht werden, um sicherzustellen, dass die Regeln wie erwartet funktionieren.

9. Verwenden der neuen Datei
    - Du kannst entweder GIT aktualisieren oder die Datei an einen Sprachbetreuer senden, wenn du dir sicher bist, dass die Qualität in Ordnung ist.

Quelle und mehr Informationen unter: http://wiki.languagetool.org/developing-robust-rules