Стеммер Портера для русского языка

Выделение основы слова это одна из задач, часто возникающая в процессе обработки текста. В школе нас конечно учили это делать, кто-нибудь помнит как? Я нет. К тому же, русский язык обладает особой сложностью и если в английском достаточно сделать несколько простых преобразований (вроде удаления ing-ового окончания) и мы уже с высокой степенью достоверности получаем требуемую основу. В русском языке эта задача намного сложней. В идеале, она решается путем составления больших морфологических словарей. Но это не путь ленивого программиста, словари надо поддерживать, ведь язык постоянно меняется, надо отдельно обрабатывать ошибки и опечатки. Путь ленивого программиста другой, специально для них, Мартин Портер в 1980 году изобрел алгоритм стемминга.  
  
Главный плюс стеммера Портера заключается в том, что он не использует никаких словарей и выделение основы осуществляется путем преобразования слова согласно определенным правилам. В интернете полно реализаций этого алгоритма, в том числе и для Java, но, к сожалению, мне не удалось найти реализацию которая прошла бы [тест самого Портера (внимание, большой документ)](http://snowball.tartarus.org/algorithms/russian/diffs.txt) и в то же время сохраняла краткость и лаконичность. Поэтому, пришлось писать велосипед основываясь на PHP версии взятой [здесь](http://forum.dklab.ru/php/advises/HeuristicWithoutTheDictionaryExtractionOfARootFromRussianWord.html)  
  
Нет смысла переписывать описание алгоритма, оно, пусть и на английском, детально изложено на [сайте](http://snowball.tartarus.org/algorithms/russian/stemmer.html)самого Мартина Портера   
  
Исходники на Java:

import java.util.regex.Matcher;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class Porter {

private static final Pattern PERFECTIVEGROUND = Pattern.compile("((ив|ивши|ившись|ыв|ывши|ывшись)|((?&lt;=[ая])(в|вши|вшись)))$");

private static final Pattern REFLEXIVE = Pattern.compile("(с[яь])$");

private static final Pattern ADJECTIVE = Pattern.compile("(ее|ие|ые|ое|ими|ыми|ей|ий|ый|ой|ем|им|ым|ом|его|ого|ему|ому|их|ых|ую|юю|ая|яя|ою|ею)$");

private static final Pattern PARTICIPLE = Pattern.compile("((ивш|ывш|ующ)|((?<=[ая])(ем|нн|вш|ющ|щ)))$");

private static final Pattern VERB = Pattern.compile("((ила|ыла|ена|ейте|уйте|ите|или|ыли|ей|уй|ил|ыл|им|ым|ен|ило|ыло|ено|ят|ует|уют|ит|ыт|ены|ить|ыть|ишь|ую|ю)|((?<=[ая])(ла|на|ете|йте|ли|й|л|ем|н|ло|но|ет|ют|ны|ть|ешь|нно)))$");

private static final Pattern NOUN = Pattern.compile("(а|ев|ов|ие|ье|е|иями|ями|ами|еи|ии|и|ией|ей|ой|ий|й|иям|ям|ием|ем|ам|ом|о|у|ах|иях|ях|ы|ь|ию|ью|ю|ия|ья|я)$");

private static final Pattern RVRE = Pattern.compile("^(.\*?[аеиоуыэюя])(.\*)$");

private static final Pattern DERIVATIONAL = Pattern.compile(".\*[^аеиоуыэюя]+[аеиоуыэюя].\*ость?$");

private static final Pattern DER = Pattern.compile("ость?$");

private static final Pattern SUPERLATIVE = Pattern.compile("(ейше|ейш)$");

private static final Pattern I = Pattern.compile("и$");

private static final Pattern P = Pattern.compile("ь$");

private static final Pattern NN = Pattern.compile("нн$");

public String stem(String word) {

word = word.toLowerCase();

word = word.replace('ё', 'е');

Matcher m = RVRE.matcher(word);

if (m.matches()) {

String pre = m.group(1);

String rv = m.group(2);

String temp = PERFECTIVEGROUND.matcher(rv).replaceFirst("");

if (temp.equals(rv)) {

rv = REFLEXIVE.matcher(rv).replaceFirst("");

temp = ADJECTIVE.matcher(rv).replaceFirst("");

if (!temp.equals(rv)) {

rv = temp;

rv = PARTICIPLE.matcher(rv).replaceFirst("");

} else {

temp = VERB.matcher(rv).replaceFirst("");

if (temp.equals(rv)) {

rv = NOUN.matcher(rv).replaceFirst("");

} else {

rv = temp;

}

}

} else {

rv = temp;

}

rv = I.matcher(rv).replaceFirst("");

if (DERIVATIONAL.matcher(rv).matches()) {

rv = DER.matcher(rv).replaceFirst("");

}

temp = P.matcher(rv).replaceFirst("");

if (temp.equals(rv)) {

rv = SUPERLATIVE.matcher(rv).replaceFirst("");

rv = NN.matcher(rv).replaceFirst("н");

}else{

rv = temp;

}

word = pre + rv;

}

return word;

}

}