NLP and Perl

Francesco Nidito
nids@di.unipi.it

May 31, 2006
Outline

Introduction

Why Perl and NLP?

Lingua::EN::*
- Parsing English
- Categorization and Extraction
- Link parsing

Conclusions
References

- http://search.cpan.org/
Caveat

- I am not a NLP expert!
I am not a NLP expert!

... but I use to have fun... ehm... work with Perl!
Natural Language Processing

- NLP...
  - ...is intended to replay to questions on human written documents like the followings:
    - what does it mean?
    - what other documents is it like?
    - ...
  - ...uses AI techniques
Perl

- Perl... is Practical Export and Report Language
  - is often described as a text processing language
  - very (very!) powerful regular expression

```perl
code:print $x if($x > 42);
```
Perl

- Perl...
  - ...is Practical Export and Report Language
  - ...is often described as a text processing language
    - very (very!) powerful regular expression
  - ...was created with natural language in mind

```perl
print $x if($x > 42);
```
Perl

- Perl...
  - ...is Practical Export and Report Language
  - ...is often described as a text processing language
    - very (very!) powerful regular expression
  - ...was created with natural language in mind
    - baby Perl
Perl

- Perl...  
  - ... is **Practical Export and Report Language**  
  - ... is often described as a **text processing** language  
    - very (very!) **powerful** **regular expression**  
  - ... was created with natural language in mind  
    - baby Perl  
    - natural language features (e.g. postfix-if)
      ```perl
      print $x if($x > 42);
      ```
Why is Perl a good choice?

- A lot of modules around
Why is Perl a good choice?

- A lot of modules around
  - CPAN (Comprehensive Perl Archive Network)
    - on line since October 26 1995 (10 years!)
    - 3061 MB
    - 283 mirrors
    - 5126 authors
    - 10066 modules
Perl NLP modules

- Lingua::*
  - “only” 337 modules
Perl NLP modules

▶ Lingua::*
  ▶ “only” 337 modules
  ▶ various languages
    ▶ english: Lingua::EN::*
    ▶ portuguese: Lingua::PT::*
    ▶ chinese: Lingua::ZH::*
Perl NLP modules

- Lingua::*
  - “only” 337 modules
  - various languages
    - english: Lingua::EN::*
    - portuguese: Lingua::PT::*
    - chinese: Lingua::ZH::*
    - klingon: Lingua::Klingon::*
Perl NLP modules

- Lingua::*
  - “only” 337 modules
  - various languages
    - english: Lingua::EN::*
    - portuguese: Lingua::PT::*
    - chinese: Lingua::ZH::*
    - klingon Lingua::Klingon::*
- utilities
  - Lingua::Stem::*
  - Lingua::Features::*
  - Lingua::StopWords::*
Perl NLP modules

- **Lingua::**: 
  - “only” 337 modules
  - various languages
    - english: Lingua::EN::*
    - portuguese: Lingua::PT::*
    - chinese: Lingua::ZH::*
    - klingon Lingua::Klingon::*
  - utilities
    - Lingua::Stem::*
    - Lingua::Features::*
    - Lingua::StopWords::*

- **Text::NLP (version 0.1)**
Perl NLP modules

- **Lingua::*
  - "only" 337 modules
  - various languages
    - english: Lingua::EN::*
    - portuguese: Lingua::PT::*
    - chinese: Lingua::ZH::*
    - klingon Lingua::Klingon::*
  - utilities
    - Lingua::Stem::*
    - Lingua::Features::*
    - Lingua::StopWords::*

- **Text::*NLP (version 0.1)**

- **NLP::*ExtractFeatures (not in CPAN)**
Lingua::EN::*

- Utility modules
  - Lingua::EN::Inflect
  - Lingua::EN::Words2Num
  - ...

Lingua::EN::*

- Utility modules
  - Lingua::EN::Inflect
  - Lingua::EN::Words2Num
  - ...

- NLP modules
  - Lingua::EN::Sentence
  - Lingua::EN::StopWords
  - Lingua::Stem::En
  - Lingua::EN::Tagger
  - Lingua::EN::Summarize
  - Lingua::EN::NamedEntitiy
  - ...

Splitting Up Sentences

- Text: Sentence
  - quite good module
  - problems with abbreviations
Splitting Up Sentences

- Text::Sentence
  - quite good module
  - problems with abbreviations
  - for instance “I do not work with abbreviations, e.g., This one.”
    - “I do not work with abbreviations, e.g.,”
    - “ This one.”
Splitting Up Sentences

- **Text::Sentence**
  - quite good module
  - problems with abbreviations
  - for instance “I do not work with abbreviations, e.g., This one.”
    - “I do not work with abbreviations, e.g.,”
    - “This one.”

- **Lingua::EN::Sentence**
  - resolves the punctuation problem
Splitting Up Sentences

- **Text::Sentence**
  - quite good module
  - problems with abbreviations
  - for instance “I do not work with abbreviations, e.g., This one.”
    - “I do not work with abbreviations, e.g.,”
    - “This one.”

- **Lingua::EN::Sentence**
  - resolves the punctuation problem
  - it uses a regular expression to cuts viciously the text into sentences
  - then a list of rules is applied on the marked text in order to fix end-of-sentence markings on places which are not indeed end-of-sentence
Splitting Up Sentences

▶ Text::Sentence
  ▶ quite good module
  ▶ problems with abbreviations
  ▶ for instance “I do not work with abbreviations, e.g., This one.”
    ▶ “I do not work with abbreviations, e.g.,”
    ▶ “This one.”

▶ Lingua::EN::Sentence
  ▶ resolves the punctuation problem
  ▶ it uses a regular expression to cuts viciously the text into sentences
  ▶ then a list of rules is applied on the marked text in order to fix end-of-sentence markings on places which are not indeed end-of-sentence
  ▶ customizable module (via add_acronyms method)
This punctuation-based assumption is generally good enough, but screws massily on sentences containing abbreviations followed by capital letters, e.g., This one. Shlomo Yona’s Lingua::EN::Sentence does a considerably better job:

my $sentences = get_sentences($text);

foreach my $sentence (@$sentences){
  printf "#",$sentence,"\n\n";
}
This punctuation-based assumption is generally good enough, but screws massively on sentences containing abbreviations followed by capital letters, e.g., This one.

Shlomo Yona’s Lingua::EN::Sentence does a considerably better job:

[nids@vultus NLP]%
Stemming and Stopwording

- **Lingua::Stem::En**
  - **Stemming** consist in reduction of words to simpler form
    - useful in building histograms
    - "volcano erupting", "volcanoes erupted" ⇒ "volcano erupt"
    - rules applied via regular expressions
Stemming and Stopwording

- **Lingua::Stem::En**
  - **Stemming** consist in reduction of words to simpler form
    - useful in building histograms
    - "volcano erupting", "volcanoes erupted" ⇒ "volcano erupt"
    - rules applied via regular expressions

- **Lingua::EN::StopWords**
  - **Stopwords** are words that do not carry semantic content
    - "are" "that" "very"…are stopwords
Stemming and Stopwording

- **Lingua::Stem::En**
  - **Stemming** consist in reduction of words to simpler form
    - useful in building **histograms**
    - "volcano erupting", "volcanoes erupted" ⇒ "volcano erupt"
    - implementation of the **Porter Stemming Algorithm** (Porter, M.F., "An Algorithm For Suffix Stripping", Program 14 (3), July 1980, pp. 130-137)
    - rules applied via **regular expressions**

- **Lingua::EN::StopWords**
  - **Stopwords** are words that do not carry semantic content
    - "are" "that" "very"...are stopwords

- **Plucene**, a Perl-based, and highly customizable search engine toolkit, based on the Lucene API, uses **Lingua::Stem::EN**
use Lingua::EN::StopWords qw(%StopWords);
use Lingua::Stem::En;
use Lingua::EN::Splitter qw(words);
use List::Util qw(sum);

print compare("The AD 79 volcanic eruption of Mount Vesuvius",
               "The volcano, Mount Vesuvius, erupted in 79AD");

print "\n";
Stemming and Stopwording (example 2/2)

sub sentence2hash{
    my $words = words(lc(shift));
    my $stemmed = Lingua::Stem::En::stem({
        -words => [grep { !$StopWords{$_} } @$words]
    });
    return { map { $_ => 1 } grep $_, @$stemmed };}

sub compare{
    my ($h1, $h2) = map { sentence2hash($_) } @_; 
    my %composite = %$h1;
    %composite{$_}++ for keys %$h2;
    return 100*(sum(values %composite)/keys %composite)/2;
}
Stemming and Stopwording (example output)

[nids@vultus NLP]% compare_ex.pl
79
[nids@vultus NLP]%

Only **23** lines of code (considering empty lines)
Bayesian Analysis

- Algorithm: NaiveBayes
  - "Naive Bayes" machine learning algorithm
  - If compared to other algorithms (kNN, SVM, Decision Trees)
    - Pretty fast
    - Reasonably competitive in quality
use XML::RSS;
use Lingua::EN::StopWords qw(%StopWords);
use Lingua::EN::Splitter qw(words);
use Algorithm::NaiveBayes;

my $nb = new NaiveBayes->new();
for my $category (qw(interesting boring)){
  my $rss = new XML::RSS;
  $rss->parsefile($category.".rdf");
  for $i (@{$rss->{items}}){
    $nb->add_instance(
      attribute => invert_item($i),
      label => $category);
  }
}

$nb->train;
sub invert_string{
    my ($string, $weight, $hash) = @_; 
    for my $i (grep {!$StopWords{$_}} @{words(lc($string))}){
        $hash->{$i} += $weight 
    }
}

sub invert_item{
    my $item = shift; 
    my %hash; 
    invert_string($item->{title}, 2, %hash); 
    invert_string($item->{description}, 1, %hash); 
    return %hash;
}
my $target = new XML::RSS;
$target->parsefile("incoming.rdf");
for my $item (@{$target->{items}}){
    print "$item->{'title'}": ";
    
    my $predict = $nb->predict(attribute => invert_item($item));
    print int($predict->{interesting}*100)."% interesting\n";
}
Bayesian Analysis (example output)

[nids@vultus NLP]% bayesian_ex.pl
Elektro, the oldest U.S. Robot: 12% interesting
Open-Source technique for GM Crops: 99% interesting
...

Only 38 lines of code (considering empty lines)
Link parsing

- **Lingua::LinkParser**
  - implements the Link Grammar Parser by Sleator, Temperley and Lafferty at CMU.
  - given a sentence, the module assigns to it a syntactic structure, which consists of set of labeled links connecting pairs of words.
  - it can be used/tested on-line (via cgi):
    http://www.link.cs.cmu.edu/link/submit-sentence-4.html
  - starting from version 4.0 supports morpho-guessing (marked with the ![] symbol)
use Lingua::LinkParser;

my $parser = new Lingua::LinkParser;
my $text = "Moses supposes his toses are roses.";

my $sentence = $parser->create_sentence($text);
my $linkage = $sentence->linkage(1);

print $parser->get_diagram($linkage);

Caveat: this code on my machine produces a segmentation fault error. But my machine is a mess :)

LEFT-WALL Moses supposes. He thought they were roses.
Conclusions

- Perl and NLP sounds good!
- A lot of tools
- Programming in Perl is not difficult after all
Conclusions

- Perl and NLP sounds good!
- A lot of tools
- Programming in Perl is not difficult after all
  - ...if something is missing you can program it by yourself
Call for contribution

- Module Lingua::IT::* is quite small
  - Lingua::IT::Conjugate
  - Lingua::IT::Hyphenate
  - Lingua::IT::Numbers
- Also other Italian Language modules are few
  - Lingua::Stem::It
  - Lingua::StopWords::IT
Call for contribution

- Module Lingua::IT::* is quite small
  - Lingua::IT::Conjugate
  - Lingua::IT::Hyphenate
  - Lingua::IT::Numbers
- Also other Italian Language modules are few
  - Lingua::Stem::It
  - Lingua::StopWords::IT
- If you work in NLP research you can contribute...
Thanks and more...

- Thank you for your attention
Thanks and more...

- Thank you for your attention
- Do you want to learn more about Perl?
Thanks and more...

- Thank you for your attention
- Do you want to learn more about Perl?
  - June 22-23 - 3rd Italian Perl Workshop
Thanks and more…

- Thank you for your attention
- Do you want to learn more about Perl?
  - June 22-23 - 3rd Italian Perl Workshop
  - This year IPW is free (yes, free as in “free beer”)

http://www.perl.it/
Thanks and more...

- Thank you for your attention
- Do you want to learn more about Perl?
  - June 22-23 - 3rd Italian Perl Workshop
  - This year IPW is free (yes, free as in “free beer”)
  - [http://www.perl.it/](http://www.perl.it/)