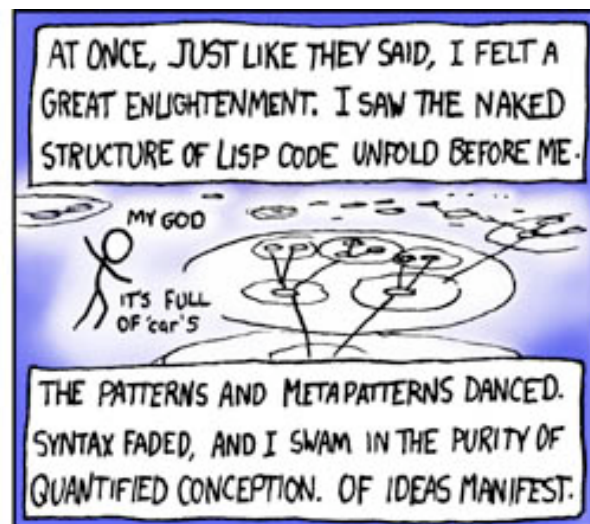# Natural Language Processing in Perl

Walt Mankowski (walt@cs.drexel.edu)
Drexel University

PPW 2007
13 October 2007

What is Natural Language Processing?

- Automated generation and understanding of human languages (English, French, Chinese, Klingon, etc.)

- Higher-level understanding of text as something more than just a string of characters.

Fun with Perl and NLP

Perl should be a great language for doing NLP:

- great at manipulating text

- powerful regular expressions

- it's functional

- Larry Wall is a linguist!

NLP modules on CPAN

An embarrassment of riches:

- 446 modules under Lingua::*

- 50 modules under Lingua::EN::*

- 3 modules under Lingua::Klingon::*

- 4 modules under Acme::Lingua::*

NLP modules on CPAN

But of course, since it's CPAN...

- not every module under Lingua::* is an NLP module

- not all NLP modules are under Lingua::*

- some modules great, others not so much

Talk Outline

Overview of NLP modules on CPAN:

- Best of Lingua::*

- Some assorted other NLP modules

- Lingua::LinkParser::*

- WordNet::*

Caveats

- I'm not a linguist

- I'm not an NLP expert

- I do need to do language processing for my research

- I can't cover every NLP module on CPAN

- I can't cover every feature for each of these modules

Lingua::EN::Conjugate

Conjugations and contractions of
English verbs

```
print conjugate( 'verb'=>'look',
                 'tense'=>'perfect_prog',
                 'pronoun'=>'he',
                 'negation'=>'not' );

# he has not been looking
```

Lingua::EN::Conjugate

Conjugations and contractions of English verbs
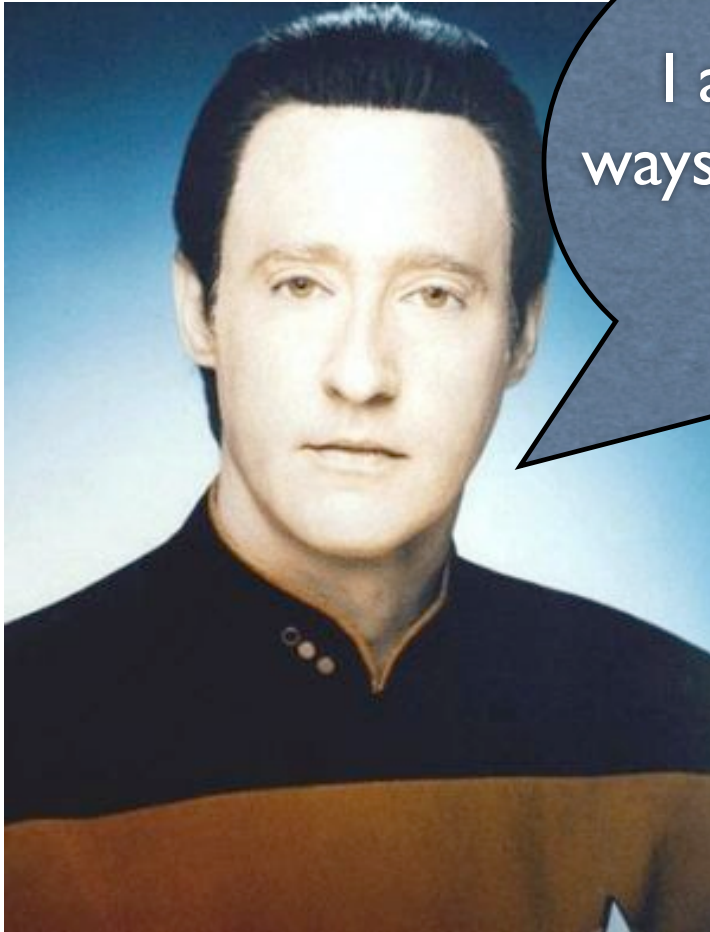
```
print contraction("I am superior, Sir, in
many ways.  But I would gladly give it up
to be human.");

# I'm superior, Sir, in many ways. But I'd
gladly give it up to be human.
```

Lingua::EN::Fathom

Measures readability of English text

```
use Lingua::EN::Fathom;

my $text = new Lingua::EN::Fathom;
$text->analyse_file("gpl.txt");
print $text->report;
```

Lingua::EN::Fathom

```
File name                  : gpl.txt
Number of characters       : 12263
Number of words            : 2016
Percent of complex words   : 15.87
Average syllables per word : 1.6334
Number of sentences        : 72
Average words per sentence : 28.0000
Number of text lines       : 200
Number of blank lines      : 48
Number of paragraphs       : 49

READABILITY INDICES

Fog                        : 17.5492
Flesch                     : 40.2266
Flesch-Kincaid             : 14.6045
```

Lingua::EN::Fathom

```
File name               : gpl.txt
Number of characters    : 12263
Number of words         : 2016
Percent of complex words : 15.87
Average syllables per word : 1.6334
Number of sentences     : 72
Average words per sentence : 28.0000
Number of text lines    : 200
Number of blank lines   : 48
Number of paragraphs    : 49


READABILITY INDICES

Fog                     : 17.5492
Flesch                  : 40.2266
Flesch-Kincaid          : 14.6045
```

Lingua::EN::FindNumber

Locates written numbers in English text

```
use Lingua::EN::FindNumber;

my $text = "Fourscore and seven years ago, our
four fathers...";
$text = numify($text);

# 87 years ago, our 4 fathers...
```

Numbers to words

- Lingua::EN::Numbers

- Lingua::EN::Numbers::Easy

- Lingua::EN::Nums2Words

- Lingua::Num2Word

Lingua::EN::Inflect

Converts singular to plural.

Written by Damian Conway.

```perl
use Lingua::EN::Inflect qw(PL);

print "The plural of ", $word, " is ", PL($word),
"\n";

print $line_count, PL(" line", $line_count),
      " updated\n";
```

Acme::Lingua::EN:: Inflect:Modern

Modernizes Lingua::EN::Inflect rule's

- Lingua::EN::Inflect

  - `2 lines updated`

- Acme::Lingua::EN::Inflect::Modern

  - `2 line's updated`

Lingua::EN::NameCase

Fixes the case of people's names

ROETHLISBERGER	Roethlisberger
mcnabb	McNabb
O'REILLY	O'Reilly
JEAN-LUC	Jean-Luc
paul vi	Paul VI

Lingua::EN::NameCase

```
use Lingua::EN::NameCase 'NameCase';

$FixedCasedName  = NameCase($OriginalName);
@FixedCasedNames = NameCase(@OriginalNames);
```

Lingua::EN::Sentence

Split text into sentences

Can use list of abbreviations

```
use Lingua::EN::Sentence qw( get_sentences
                             add_acronyms );

# add support for 'Lt. Gen.'
add_acronyms('lt','gen');

# get the sentences
my $sentences=get_sentences($text);
```

Text::Sentence

- No support for abbreviations

- Use Lingua::EN::Sentence instead

Lingua::EN::Squeeze

Shortens text to minumum syllables

Lingua::EN::Squeeze

Before:

You can use this module e.g. to preprocess text before it is sent to electronic media that has some maximum text size limit. For example pagers have an arbitrary text size limit, typically around 200 characters, which you want to fill as much as possible. Alternatively you may have GSM cellular phone which is capable of receiving Short Messages (SMS), whose message size limit is 160 characters. For demonstration of this module's SqueezeText() function, this paragraph's conversion result is presented below. See yourself if it's readable (Yes, it takes some time to get used to). The compression ratio is typically 30-40%

After:

u _n use thi mod e.g. to prprce txt bfre i_s snt to elrnic mda has som max txt siz lim. f_xmple pag hv abitry txt siz lim, tpcly 200 chr, W/ u wnt to fll as mch as psbleAlternatvly u may hv GSM cllar P8 w_s cpble of rcivng Short msg (SMS), WS/ msg siz lim is 160 chr. 4 demonstrton of thi mods SquezText fnc , dsc txt of thi prgra has ben cnvd_ blow See uself if i_s redble (Yes, it tak som T to get usdto compr rat is tpcly 30-40

Lingua::StopWords

Typical stop words for 12 different languages

```
my $stopwords = getStopWords('en');
print join ' ',
      grep { !$stopwords->{$_} } @words;
```

Also Lingua::EN::StopWords

Lingua::EN::Syllable

Estimates syllable count in words

```
$count =
  syllable('supercalifragilisticexpialidocious');

# 14
```

Claims to work 85-95% of the time, but doesn't do dictionary lookups

Lingua::EN::Tagger

Part-of-speech tagger for English

```
my $p = new Lingua::EN::Tagger;
my $text = "the quick brown fox jumped over the lazy
dog";
my $tagged_text = $p->add_tags( $text );


<det>the</det> <jj>quick</jj> <jj>brown</jj>
<nn>fox</nn> <vbd>jumped</vbd> <in>over</in>
<det>the</det> <jj>lazy</jj> <nn>dog</nn>
```

Lingua::Atinlay::Ingpay

Erlpay Odulemay otay Onvertcay
Englishhay otay Igpay Atinlay

Ittenwray ybay Aseycay Estway

```
use Lingua::Atinlay::Igpay qw/:all/;

my $text = "the quick brown fox jumped over the lazy
dog";
print enhay2igpayatinlay($text), "\n";

# ethay uickqay ownbray oxfay umpedjay overhay ethay
azylay ogday
```

Lingua::Conjunction

Convert perl lists into linguistic conjunctions

```
$name_list = conjunction('Jack', 'Jill', 'Spot');
# " Jack, Jill, and Spot"


Lingua::Conjunction->penultimate(0);
$name_list = conjunction('Jack', 'Jill', 'Spot');
# " Jack, Jill and Spot"


$name_list = conjunction('Jack, a boy', 'Jill, a
girl', 'Spot, a dog');
# "Jack, a boy; Jill, a girl; and Spot, a dog"
```

Lingua::Identify

Guesses most probable language for text

```perl
use Lingua::Identify
  qw(:language_identification);

# scalar contex -- most probable
my $lang = langof($txt);

# list context -- pairs of langs/probs
my @lang = langof($txt);
```

Lingua::Identify

Tested on gpl.txt:

English	26.7%
French	6.7%
Romanian	4.3%
Turkish	0.7%

Lingua::Stem

Stemming of words

```
use Lingua::Stem qw/stem/;

my $text = "the quick brown foxes jumped over the
lazy dogs";
my @words = split / /, $text;
my $stems = stem(@words);
print "@$stems\n";

# the quick brown fox jump over the lazi dog
```

Lingua::LinkParser

Link Grammar

- Theory of English syntax

- Parses sentence into a set of labeled links connecting pairs of words

- Developed at CMU

- http://www.link.cs.cmu.edu/link/

Link Grammar

- Think of words as blocks or a model train

- Connectors can come out of words to the left or right

- Many types of connectors (over 100)

- A left connector on one word must join with a right connector of the same type on another word

```
    +-----Ds-----+
    |        +---A--+-Ss-+-PP-+
    |        |      |    |    |
   the black.a dog.n has gone
```

- Ds – determiner to singular noun

- A – adjective to noun

- Ss – singular subject to verb

- PP – forms of "have" to past participles

Installation

- API doesn't have an install script

- Debian package, but Lingua::LinkParser can't use it

  - expects includes and objs under same directory

- Suggestion: just keep API in your homedir if you're testing it

Using Lingua::LinkParser

```perl
use Lingua::LinkParser;

my $parser = new Lingua::LinkParser;
my $txt = "the quick brown fox jumped over the lazy dog";
my $sentence = $parser->create_sentence($txt);
my @linkages = $sentence->linkages;
for my $link (@linkages) {
  ...
}
```

get_diagram()

```
    +---------Ds---------+                        +-------Js------+
    |       +-------A------+               |      +-----Ds----+
    |       |       +---A--+---Ss--+--MVp-+  |      +--A--+
    |       |       |      |      |      |  |      |     |
the quick.a brown.a fox.n jumped.v over the lazy.a
dog.n


    +---------Ds---------+                 +----------Osn---------+
    |       +-------A------+               |            +-----Ds----+
    |       |       +---A--+---Ss--+---K--+  |      +--A--+
    |       |       |      |      |      |  |      |     |
the quick.a brown.a fox.n jumped.v over the lazy.a
dog.n
```

print_constituent_tree()

```
(S (NP the quick brown fox)
   (VP jumped
       (PP over
           (NP the lazy dog))))

[S [NP the quick brown fox NP] [VP jumped [PP over [NP
the lazy dog NP] PP] VP] S]



(S (NP the quick brown fox)
   (VP jumped
       (PRT over)
       (NP the lazy dog)))

[S [NP the quick brown fox NP] [VP jumped [PRT over PRT]
[NP the lazy dog NP] VP] S]
```

words

```
for my $link (@linkages) {
    my @words = $link->words;
    for my $word (@words) {
        print $word->text, "\n";
    }
}
```

```
LEFT-WALL
the
quick.a
brown.a
fox.n
jumped.v
over
the
lazy.a
dog.n
RIGHT-WALL
```
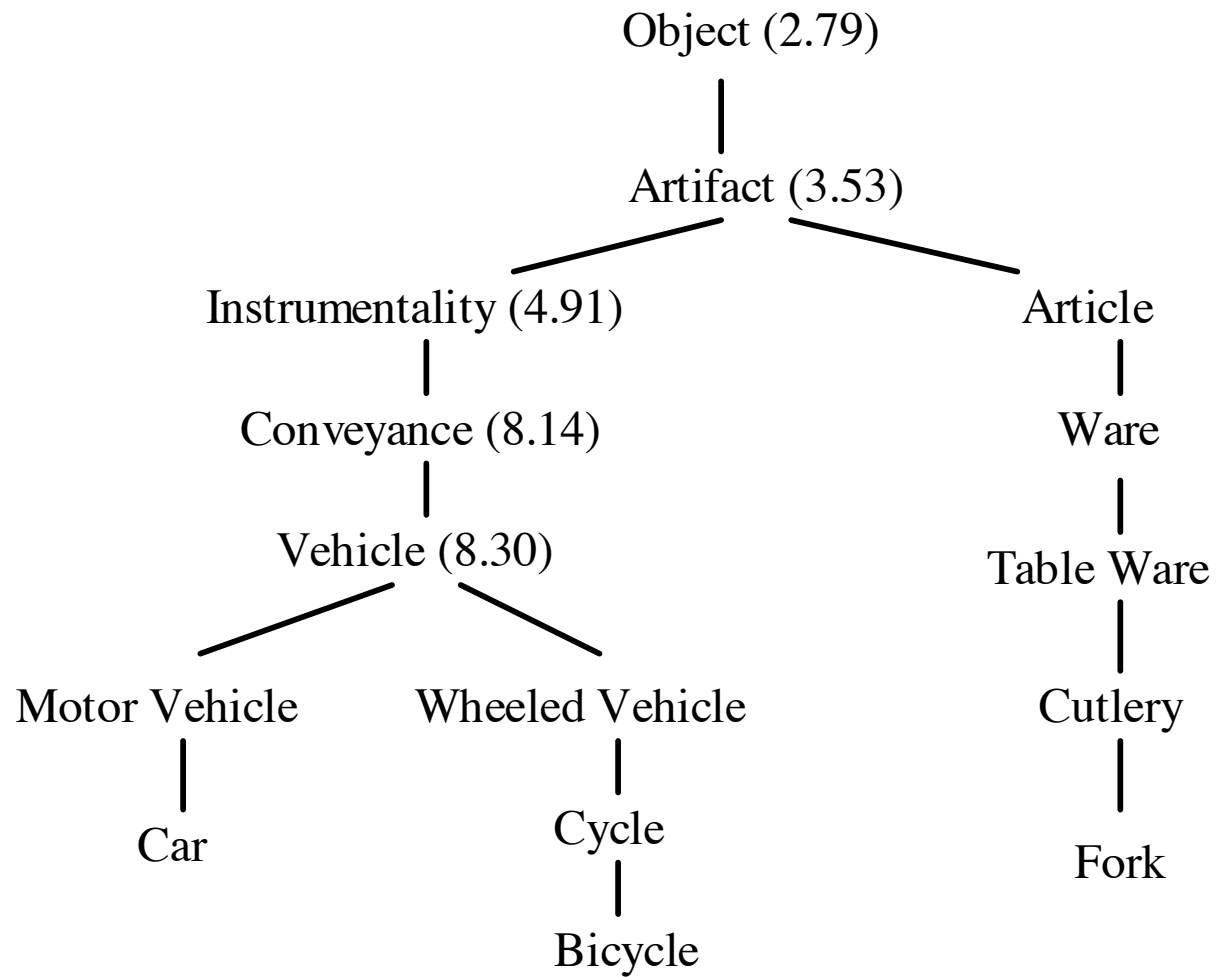
WordNet

WordNet

- Large lexical database of English

- Words stored hierarchically from general to specific

- Only nouns, verbs, adjectives, and adverbs

- Over 19,000 papers on Google Scholar

- http://wordnet.princeton.edu/

WordNet::Similarity

- Computes semantic similarity between words using WordNet

- About a dozen different measures implemented

- I used method by Jiang and Conrath (WordNet::Similarity::jcn)

- Research to decide what's best for you

Object (2.79)

Artifact (3.53)

Instrumentality (4.91)   Article

Conveyance (8.14)   Ware

Vehicle (8.30)   Table Ware

Motor Vehicle   Wheeled Vehicle   Cutlery

Car   Cycle   Fork

Bicycle

Jiang & Conrath, 1997

Installation

- WordNet 2.1

    - **not** WordNet 3.0

- WordNet::QueryData 1.45

    - **not** WordNet::QueryData 1.46

- WordNet::Similarity 1.04

- `export WNHOME=/usr/share/wordnet`
  (if using Debian package)

```perl
#!/usr/local/bin/perl -w
use strict;
use WordNet::Similarity::path;
use WordNet::QueryData;

my $wn = new WordNet::QueryData;
my $rel = new WordNet::Similarity::path($wn);

print $rel->getRelatedness("car#n#1", "bus#n#1");
print "\t";
print $rel->getRelatedness("car#n#1", "bus#n#2");
print "\n";

# 0.125   0.0476190476190476
```

```perl
#!/usr/local/bin/perl -w
use strict;
use WordNet::Similarity::jcn;
use WordNet::QueryData;

my $wn = new WordNet::QueryData;
my $rel = new WordNet::Similarity::jcn($wn);

print $rel->getRelatedness("car#n#1", "bus#n#1");
print "\t";
print $rel->getRelatedness("car#n#1", "bus#n#2");
print "\n";

# 0.145512180634563        0
```

word#pos#sense

word

- base form (stem)

pos

- n=noun, v=verb, a=adjective, r=adverb

sense

- sense word is being used it

- numeric

wn->validForms()

```
print $wn->validForms("bus");
# bus#n, bus#v

print $wn->validForms("made");
# make#v, made#a

print $wn->validForms("made#v");
# make#v
```

wn->querySense()

```
print $wn->querySense("bus#n");
# bus#n#1, bus#n#2, bus#n#3, bus#n#4

print $wn->querySense("make#v");
# 49 senses
```

Summary

- lots of NLP modules on cpan

- I've only scratched the surface

- nothing work perfectly

- parsing English is HARD

Thanks!