Przetwarzanie języka w praktyce, na przykładzie problemu rozstrzygania wieloznaczności

Aleksander Pohl
http://apohllo.pl

Katedra Informatyki, Akademia Górniczo-Hutnicza

SFI 8. marca 2008

Plan prezentacji

Określenie celu

Wymagania

Ferret

Wieloznaczność

Cel

Podstawowy cel:

- Obsługa fleksji w napisanej w Ruby bibliotece **Ferret**, będącej portem Apache Lucene.

Po co?

- Wyszukiwanie pełnotekstowe na polskojęzycznych stronach WWW

Coś więcej?

- Zobaczymy... ;-)

Cel

Podstawowy cel:

- Obsługa fleksji w napisanej w Ruby bibliotece **Ferret**, będącej portem Apache Lucene.

Po co?

- Wyszukiwanie pełnotekstowe na polskojęzycznych stronach WWW

Coś więcej?

- Zobaczymy... ;-)

Cel

Podstawowy cel:

- Obsługa fleksji w napisanej w Ruby bibliotece **Ferret**, będącej portem Apache Lucene.

Po co?

- Wyszukiwanie pełnotekstowe na polskojęzycznych stronach WWW

Coś więcej?

- Zobaczymy... ;-)

Wymagania

- Interpreter **Rubiego** (lub Pythona): `ruby-lang.org`
- Biblioteka **Morfeusz** (alternatywnie: CLP, fsa_morph): `nlp.ipipan.waw.pl/~wolinski/morfeusz/morfeusz.html`
- gem **RubyInline** (dla Pythona - niepotrzebne)
- gem **ferret** (dla Pythona - lupy)
- binding **Ruby - Morfeusz**: `apohllo.pl/src/morfeusz.rb`
- binding **Python - Morfeusz**: `students.mimuw.edu.pl/~jw209508/nlp/morfeusz.py`

Wymagania

- Interpreter **Rubiego** (lub Pythona): `ruby-lang.org`
- Biblioteka **Morfeusz** (alternatywnie: CLP, fsa_morph): `nlp.ipipan.waw.pl/~wolinski/morfeusz/morfeusz.html`
- gem **RubyInline** (dla Pythona - niepotrzebne)
- gem **ferret** (dla Pythona - lupy)
- binding **Ruby - Morfeusz**: `apohllo.pl/src/morfeusz.rb`
- binding **Python - Morfeusz**: `students.mimuw.edu.pl/~jw209508/nlp/morfeusz.py`

Wymagania

- Interpreter **Rubiego** (lub Pythona): `ruby-lang.org`
- Biblioteka **Morfeusz** (alternatywnie: CLP, fsa_morph): `nlp.ipipan.waw.pl/~wolinski/morfeusz/morfeusz.html`
- gem **RubyInline** (dla Pythona - niepotrzebne)
- gem **ferret** (dla Pythona - lupy)
- binding **Ruby - Morfeusz**: `apohllo.pl/src/morfeusz.rb`
- binding **Python - Morfeusz**: `students.mimuw.edu.pl/~jw209508/nlp/morfeusz.py`

Wymagania

- Interpreter **Rubiego** (lub Pythona): `ruby-lang.org`
- Biblioteka **Morfeusz** (alternatywnie: CLP, fsa_morph): `nlp.ipipan.waw.pl/~wolinski/morfeusz/morfeusz.html`
- gem **RubyInline** (dla Pythona - niepotrzebne)
- gem **ferret** (dla Pythona - lupy)
- binding **Ruby - Morfeusz**: `apohllo.pl/src/morfeusz.rb`
- binding **Python - Morfeusz**: `students.mimuw.edu.pl/~jw209508/nlp/morfeusz.py`

Wymagania

- Interpreter **Rubiego** (lub Pythona): `ruby-lang.org`
- Biblioteka **Morfeusz** (alternatywnie: CLP, fsa_morph): `nlp.ipipan.waw.pl/~wolinski/morfeusz/morfeusz.html`
- gem **RubyInline** (dla Pythona - niepotrzebne)
- gem **ferret** (dla Pythona - lupy)
- binding **Ruby - Morfeusz**: `apohllo.pl/src/morfeusz.rb`
- binding **Python - Morfeusz**: `students.mimuw.edu.pl/~jw209508/nlp/morfeusz.py`

Indeksu

```
1:   require 'ferret'
2:   include Ferret
3:   index = Index::Index.new(:path => "./index")
4:   index << {:id => 1, :content => "Ala ma kota"}
5:   index << {:id => 2, :content => "To kot a to Ala"}
6:   index.flush
5:   index.search_each("Ala") do |id,score|
7:     puts "Dokument #{id}: #{score}"
8:     highlights = index.
9:       highlight(query,id, :field => :content,
10:                :pre_tag => "\033[36m",
11:                :post_tag => "\033[m")
12:    puts highlights
12:  end
```

Indeksu

```
1:   require 'ferret'
2:   include Ferret
3:   index = Index::Index.new(:path => "./index")
4:   index << {:id => 1, :content => "Ala ma kota"}
5:   index << {:id => 2, :content => "To kot a to Ala"}
6:   index.flush
5:   index.search_each("Ala") do |id,score|
7:     puts "Dokument #{id}:  #{score}"
8:     highlights = index.
9:       highlight(query,id, :field => :content,
10:                  :pre_tag => "\033[36m",
11:                  :post_tag => "\033[m")
12:    puts highlights
12:  end
```

Indeksu

```
1:   require 'ferret'
2:   include Ferret
3:   index = Index::Index.new(:path => "./index")
4:   index << {:id => 1, :content => "Ala ma kota"}
5:   index << {:id => 2, :content => "To kot a to Ala"}
6:   index.flush
5:   index.search_each("Ala") do |id,score|
7:     puts "Dokument #{id}:  #{score}"
8:     highlights = index.
9:       highlight(query,id, :field => :content,
10:                :pre_tag => "\033[36m",
11:                :post_tag => "\033[m")
12:    puts highlights
12:  end
```

Indeksu

```
1:   require 'ferret'
2:   include Ferret
3:   index = Index::Index.new(:path => "./index")
4:   index << {:id => 1, :content => "Ala ma kota"}
5:   index << {:id => 2, :content => "To kot a to Ala"}
6:   index.flush
5:   index.search_each("Ala") do |id,score|
7:     puts "Dokument #{id}:  #{score}"
8:     highlights = index.
9:       highlight(query,id, :field => :content,
10:                  :pre_tag => "\033[36m",
11:                  :post_tag => "\033[m")
12:    puts highlights
12:  end
```

Analizator

```
1:  class PolishLexer < Analysis::TokenStream
2:  end
3:  class MyAnalyzer < Analysis::Analyzer
4:    def token_stream(field, str)
5:      return PolishLexer.new(
6:        Analysis::LowerCaseFilter.new(
7:          Analysis::StandardTokenizer.new(str))))
8:    end
9:  end
10: index = Index::Index.new(:path => "./index",
11:                          :analyzer => MyAnalyzer.new)
```

Analizator

```
1:   class PolishLexer < Analysis::TokenStream
2:   end
3:   class MyAnalyzer < Analysis::Analyzer
4:     def token_stream(field, str)
5:       return PolishLexer.new(
6:         Analysis::LowerCaseFilter.new(
7:           Analysis::StandardTokenizer.new(str))))
8:     end
9:   end
10:  index = Index::Index.new(:path => "./index",
11:                :analyzer => MyAnalyzer.new)
```

Analizator

```
1:   class PolishLexer < Analysis::TokenStream
2:   end
3:   class MyAnalyzer < Analysis::Analyzer
4:     def token_stream(field, str)
5:       return PolishLexer.new(
6:         Analysis::LowerCaseFilter.new(
7:           Analysis::StandardTokenizer.new(str))))
8:     end
9:   end
10:  index = Index::Index.new(:path => "./index",
11:                :analyzer => MyAnalyzer.new)
```

Lexer

```
1:   class PolishLexer < Analysis::TokenStream
2:     def initialize(stream)
3:       @stream = stream
4:     end
5:     def next
6:       @stream.next
7:     end
8:     def text=(content)
9:       @stream.text = content
10:    end
11:  end
```

Lexer – next

```
1:   class PolishLexer < Analysis::TokenStream
2:     def next
3:       token = @stream.next
4:        if !token.nil?
5:         deflected = deflect(token.text)
6:         case deflected
7:           when String
8:             token.text = deflected
9:           when Array
10:            token.text = deflected[0]
11:          end
12:      end
13:      token
14:    end
15:  end
```

Lexer – deflect

```
1:   Lexeme = Apohllo::Morfeusz::Lexeme
2:   class PolishLexer < Analysis::TokenStream
3:     private
4:     def deflect(text)
5:       lexemes = Lexeme.find(text)
6:       if lexemes.size > 0
7:         lexemes.collect{|l| l.base_form}.uniq
8:       else
9:         text
10:      end
11:    end
12:  end
```

Czy to rozwiązanie jest zadowalające?

»Nasza reprezentacja zdobyła sześć goli.«

- *nasza*: 2 formy podstawowe: nasz, naszać
- *goli*: 3 formy podstawowe: gol, golić, goły

W naszym rozwiązaniu brana jest tylko pierwsza forma podstawowa!

```
when Array :  token.text = deflected[0]
```

Rozwiązanie: umieścić w indeksie wszystkie formy podstawowe.

Wszystkie formy

```
1:   class PolishLexer < Analysis::TokenStream
2:     def next
3:       unless @my_tokens.nil? or @my_tokens.empty?
4:         return @my_tokens.shift
5:       end
6:         case deflected
7:           when Array
8:             token.text = deflected[0]
9:             @my_tokens = deflected.
10:              collect{|d| Analysis::
11:                Token.new(d,token.start,token.end)}
12:        end
13:    end
14:  end
```

Czy to już koniec?

- Żeby rozwiązanie działało poprawnie, konieczne byłoby modyfikowanie również zapytania tak, aby uwzględniało ten specyficzny przypadek.
- W istocie częściowo jest to zapewnione – jeśli wprowadzimy inną formę, która posiada tę samą formę podstawową, to wszystko dzieje się właściwie.
- Jednak w przypadku gdy podamy dokładnie tę formę – *goli*, nie otrzymamy wyników gdzie będzie występowało słowo *gol*, *golił*, ani *goły*.
- A może powinniśmy inaczej podejść do problemu? Przecież tylko jedna forma podstawowa jest poprawna: *gol*! Ale jak ją wybrać?

Czy to już koniec?

- Żeby rozwiązanie działało poprawnie, konieczne byłoby modyfikowanie również zapytania tak, aby uwzględniało ten specyficzny przypadek.
- W istocie częściowo jest to zapewnione – jeśli wprowadzimy inną formę, która posiada tę samą formę podstawową, to wszystko dzieje się właściwie.
- Jednak w przypadku gdy podamy dokładnie tę formę – *goli*, nie otrzymamy wyników gdzie będzie występowało słowo *gol*, *golił*, ani *goły*.
- A może powinniśmy inaczej podejść do problemu? Przecież tylko jedna forma podstawowa jest poprawna: *gol*! Ale jak ją wybrać?

Czy to już koniec?

- Żeby rozwiązanie działało poprawnie, konieczne byłoby modyfikowanie również zapytania tak, aby uwzględniało ten specyficzny przypadek.
- W istocie częściowo jest to zapewnione – jeśli wprowadzimy inną formę, która posiada tę samą formę podstawową, to wszystko dzieje się właściwie.
- Jednak w przypadku gdy podamy dokładnie tę formę – *goli*, nie otrzymamy wyników gdzie będzie występowało słowo *gol*, *golił*, ani *goły*.
- A może powinniśmy inaczej podejść do problemu? Przecież tylko jedna forma podstawowa jest poprawna: *gol*! Ale jak ją wybrać?

Czy to już koniec?

- Żeby rozwiązanie działało poprawnie, konieczne byłoby modyfikowanie również zapytania tak, aby uwzględniało ten specyficzny przypadek.
- W istocie częściowo jest to zapewnione – jeśli wprowadzimy inną formę, która posiada tę samą formę podstawową, to wszystko dzieje się właściwie.
- Jednak w przypadku gdy podamy dokładnie tę formę – *goli*, nie otrzymamy wyników gdzie będzie występowało słowo *gol*, *golił*, ani *goły*.
- A może powinniśmy inaczej podejść do problemu? Przecież tylko jedna forma podstawowa jest poprawna: *gol*! Ale jak ją wybrać?

Możliwe rozwiązania – gramatyka 1

- Wykorzystanie zależności gramatycznych:
  »..zdobyła (*kogo? co?*) sześć (*kogo? czego?*) goli.«
- Na ostatnim miejscu oczekujemy rzeczownika w dopełniaczu. Spośród słów: *gol*, *golić*, *goli*; tylko pierwsze jest rzeczownikiem.
- **Problem**: żeby określić to wymaganie musimy wcześniej znać właściwy opis morfosyntaktyczny elementów znajdujących się w kontekście danego słowa. Zwykle opisy te nie są jednoznaczne.

Możliwe rozwiązania – gramatyka 1

- Wykorzystanie zależności gramatycznych: »..zdobyła (*kogo? co?*) sześć (*kogo? czego?*) goli.«
- Na ostatnim miejscu oczekujemy rzeczownika w dopełniaczu. Spośród słów: *gol*, *golić*, *goli*; tylko pierwsze jest rzeczownikiem.
- **Problem**: żeby określić to wymaganie musimy wcześniej znać właściwy opis morfosyntaktyczny elementów znajdujących się w kontekście danego słowa. Zwykle opisy te nie są jednoznaczne.

Możliwe rozwiązania – gramatyka 1

- Wykorzystanie zależności gramatycznych: »..zdobyła (*kogo? co?*) sześć (*kogo? czego?*) goli.«
- Na ostatnim miejscu oczekujemy rzeczownika w dopełniaczu. Spośród słów: *gol*, *golić*, *goli*; tylko pierwsze jest rzeczownikiem.
- **Problem**: żeby określić to wymaganie musimy wcześniej znać właściwy opis morfosyntaktyczny elementów znajdujących się w kontekście danego słowa. Zwykle opisy te nie są jednoznaczne.

Możliwe rozwiązania – gramatyka 2

Dla zdania:
Nie ma rzeczy bardziej zwykłej i naturalnej niż to, że ludzie, którzy mają roszczenie, iż odkryli jakąś rzecz nową w świecie filozofii i nauk, sugerują innym, by chwalili ich własne systemy, osławiając jednocześnie wszystkie te, które powstały wcześniej.
otrzymujemy **120960 kombinacj**!

Możliwe rozwiązania – semantyka

- »Nasza reprezentacja *zdobyła* sześć *goli*.«
- Zdobyć można: *Mount Everest*, *kobietę*, *gol*, ale nie *gołych* ani *golić* (to nie ma sensu!).
- Zależności semantyczne są znacznie bardziej specyficzne niż zależności gramatyczne, dlatego łatwo można zredukować liczbę możliwych interpretacji.
- **Problem**: stworzenie słownika, który zawierałby wystarczająco dokładne opisy słów – bardzo czasochłonne, nie wiadomo, kiedy opis jest wystarczająco dobry.

Możliwe rozwiązania – semantyka

- »Nasza reprezentacja *zdobyła* sześć *goli*.«
- Zdobyć można: *Mount Everest*, *kobietę*, *gol*, ale nie *gołych* ani *golić* (to nie ma sensu!).
- Zależności semantyczne są znacznie bardziej specyficzne niż zależności gramatyczne, dlatego łatwo można zredukować liczbę możliwych interpretacji.
- **Problem**: stworzenie słownika, który zawierałby wystarczająco dokładne opisy słów – bardzo czasochłonne, nie wiadomo, kiedy opis jest wystarczająco dobry.

Możliwe rozwiązania – semantyka

- »Nasza reprezentacja *zdobyła* sześć *goli*.«
- Zdobyć można: *Mount Everest*, *kobietę*, *gol*, ale nie *gołych* ani *golić* (to nie ma sensu!).
- Zależności semantyczne są znacznie bardziej specyficzne niż zależności gramatyczne, dlatego łatwo można zredukować liczbę możliwych interpretacji.
- **Problem**: stworzenie słownika, który zawierałby wystarczająco dokładne opisy słów – bardzo czasochłonne, nie wiadomo, kiedy opis jest wystarczająco dobry.

Możliwe rozwiązania – semantyka

- »Nasza reprezentacja *zdobyła* sześć *goli*.«
- Zdobyć można: *Mount Everest*, *kobietę*, *gol*, ale nie *gołych* ani *golić* (to nie ma sensu!).
- Zależności semantyczne są znacznie bardziej specyficzne niż zależności gramatyczne, dlatego łatwo można zredukować liczbę możliwych interpretacji.
- **Problem**: stworzenie słownika, który zawierałby wystarczająco dokładne opisy słów – bardzo czasochłonne, nie wiadomo, kiedy opis jest wystarczająco dobry.

Koniec

Dziękuję!