Syn!bad: A short guide

by Ovidiu Şerban

Contents

1 Introduction

Note: The purpose of this document is to explain the Syn!bad syntax among with some simple usage examples.

Syn!bad is an extended regular expression language, for usage mainly in Natural Language Processing (NLP) applications. It uses the some extended POSIX Regular Expression structures among other, more specific to NLP domain.

The name, Syn!bad (written also: Synnbad, with double nn, instead of n!) is an acronym of Synonyms [are] not bad. This would suggest that the main concepts of Syn!bad would be centred among synonyms processing, using different dictionaries.

2 Syntax

⟨*expression*⟩ ::= ⟨*token*⟩ ' ' ⟨*expression*⟩ | ⟨*token*⟩

⟨*token*⟩ ::= ⟨*pattern*⟩
| ⟨*pattern*⟩ '`*`'
| ⟨*pattern*⟩ '`?`'
| ⟨*pattern*⟩ '{' ⟨*number*⟩ '`,`' ⟨*number*⟩ '}'

⟨*pattern*⟩ ::= ⟨*word*⟩
| '`<`' ⟨*POS_Structure*⟩ '`>`'
| '`[`' ⟨*synonym*⟩ '`]`'
| '`$`' ⟨*variable*⟩

⟨*POS_Structure*⟩ ::= ⟨*POS*⟩ ('`#`' ⟨*variable*⟩)?

⟨*POS*⟩ ::= ⟨*PennPOS*⟩
| ⟨*GenericPOS*⟩

⟨*synonym*⟩ ::= ⟨*word*⟩ ('|' ⟨*POS*⟩)? ('`#`' ⟨*variable*⟩)?

⟨*number*⟩ ::= [1-9] [0-9]*

⟨*word*⟩ ::= [a-z]+

⟨*variable*⟩ ::= [a-z0-9]+

⟨*PennPOS*⟩ ::= ‘JJ’ | ‘RB’ | ‘DT’ | ‘TO’ | ‘RP’ | ‘RBR’ | ‘RBS’ | ‘LS’
| ‘JJS’ | ‘JJR’ | ‘FW’ | ‘NN’ | ‘NNPS’ | ‘VBN’ | ‘VB’ | ‘VBP’
| ‘PDT’ | ‘WP$’ | ‘PRP’ | ‘MD’ | ‘SYM’ | ‘WDT’ | ‘VBZ’ | ‘¨’
| ‘#’ | ‘WP’ | ‘’’ | ‘IN’ | ‘$’ | ‘VBG’ | ‘EX’ | ‘POS’ | ‘(’
| ‘VBD’ | ‘)’ | ‘.’ | ‘,’ | ‘UH’ | ‘NNS’ | ‘CC’ | ‘CD’ | ‘NNP’
| ‘PP$’ | ‘:’ | ‘WRB’

⟨*GenericPOS*⟩ ::= ‘#*’ | ‘VB*’ | ‘RB*’ | ‘NN*’ | ‘JJ*’

3 Example

Considering the complexity of the rules, some examples are given in order to simplify most of the logic:

- POS_Structure encapsulates a part of speech structure, given by the Penn POS Tag System or a more simple formalisation, called Generic POS.

  Also, POS_Structure offers the possibility of retrieving the matched structure as a variable, given after the # sign.

- Generic POS contains five values:

  #* which groups all the punctuation into one single category

  VB* which groups all the verb tags

  RB* which groups all the adverb tags

  NN* which groups all the noun tags

  JJ* which groups all the adjective tags

- the synonym structure, does a comparison based on synonyms:

  using the `|` some POS restictions could be decided, based on the POS rules

  Also, synonym offers the possibility of retrieving the matched structure as a variable, given after the # sign.

Based on the rules described above, this would be a valid pattern, containing most of the Syn!bad capabilities:

```
$name <#*>? do you <VB*>* [some|RB*] [water#object]
```

And given the following sentence:

```
Ovidiu , do you want some water
```

The result of the matching would be: $*name* ← *Ovidiu* and #*object* ← *water*, while `<#*>?` will match , and `<VB*>*` will match the single verb *want*.