Evolutionary Learning of Syntax Patterns for Genic Interaction Extraction

Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, Fabiano Tarlao, Marco Virgolin
Problem

➔ Identifying sentences that contain **interactions** between **genes** and **proteins**
   ◆ from biomedical literature

➔ Available data:
   ◆ dictionary of genes, proteins and interactors
   ◆ example sentences
Why?

→ Biomedical literature is:
  ◆ vast
  ◆ rapidly growing

→ Challenging problem: automatic extraction of knowledge from a text in natural language
  ◆ informations are “diluted” in the text
  ◆ very challenging problem: discover relations between entities
Goal

→ Generation of a classifier $C$ in order to identify sentences containing interactions between genes and proteins

◆ automatically

◆ based on recurring syntactic patterns
Our approach

➔ Classifier $C$ is a set of regular expressions ($regex$)

\[ C = \{ r_1, r_2, \ldots \} \]

➔ Each regex is a sentence classifier ("accepts" or "does not accept")

◆ $C$ accepts sentences accepted by at least one regex

➔ Regex applied on a semantical representation of the text
Our approach (II)

➔ Regex generated **automatically**

- by means of Genetic Programming (GP)

- starting from **examples**
  - strings which must be accepted
  - strings which must **not** be accepted
Sentences preprocessing

Mapping of a sentence $s$ in a $\phi$-string $x$

a. substitution of words in $s$ with “annotations”
   i. gene, protein, interactor
   or
   ii. Part-Of-Speech

b. mapping of annotations in Unicode characters

c. concatenation
Sentences preprocessing (II)

Example:

\[ S = YfhP \text{ may act as a negative regulator for the transcription of } yfhQ \]

\[ \Downarrow \]

\[ [YfhP] \text{ [may] [act] [as] [a] [negative] [regulator] [for] [the] [transcription] [of] [yfhQ]} \]

\[ \Downarrow \]

\[ [GENEPTN] \text{ [MD] [VB] [IN] [DT] [JJ] [INOUN] [IN] [DT] [INOUN] [IN] [GENEPTN]} \]

\[ \Downarrow \]

\[ x = GB0if6JifJiG \]
Generation of $C$: GP

- We used a Tree-based GP
- In this work
  candidate solution = regex
Key aspects

➔ Multi-objective fitness:
◆ $f = (\text{Accuracy}, \text{FPR}, \text{Regex length})$
◆ we purposefully avoided to include any problem-specific knowledge (gene/protein/...)

➔ Problem handled by mean of separate-and-conquer

➔ Final output: set of regular expressions $C = \{ r_1, r_2, ... \}$
Separate-and-conquer

➔ Each regex $r_i \in C$ makes an independent and parallel classification

➔ Each regex is tailored for a sub-problem
  ◆ the problem is solved “step-by-step”

➔ Final output = logic OR of classifications
Separate-and-conquer

- $C = \emptyset$
- we execute a GP search over the examples obtaining $r^*$
- if $FPR < \text{threshold}$
  - $C = C \cup \{r^*\}$
- else
  - terminate
- remove from the positive examples those which were classified correctly by $r^*$
Classifier example

\[ C = \{ r_1, r_2 \} \]

\[ r_1 = \text{GENEPTN}[\text{^RB}][\text{^NNS VBN GENEPTN}]++ \]

\[ r_2 = . \text{INOUN IN GENEPTN} . [\text{^DT NN}] \]
Experimental evaluation: the data

→ Dataset: 456 sentences from biomedical papers
  ◆ ½ with interactions e ½ without
  ◆ manually labelled by experts

→ Dataset splitted in Learning e Testing
  ◆ ≈80% examples in Learning
  ◆ ≈20% examples in Testing

→ 5 fold randomly generated
  ◆ with Testing_i ≠ Testing_j
Baseline 1, 2: problem specific knowledge

➔ Annotations-Co-Occurrence
◆ it is tightly tailored to this specific problem
◆ sentence is positive if contains
  ● at least 2 genes/proteins
  ● at least 1 interactor

➔ Annotations-LLL05-Patterns
◆ 10 pattern generated in “LLL'05 Challenge: Genic Interaction Extraction with Alignments and Finite State Automata”
  - J. Hakenberg et alia
◆ built over >90% of the dataset
  (also testing!)
Baseline 3: $\phi$-SSLEA

- Based on *Smart State Labeling Algorithm*
  - algorithm for *DFA learning*
  - works well in presence of noise

- **Hill-Climbing**

- Generates **DFA** which accepts or refuse a $\phi$-string $x$
  - if $x$ accepted $\Rightarrow x$ contains an interaction between gene/protein
  - otherwise, no
Baseline 4, 5: *Words-NaiveBayes e Words-SVM*

→ Standard for text classification
  ◆ Supervised Machine Learning methods

→ Feature based on *word occurrences*

→ Preprocessing
  ◆ stemming
  ◆ features selection
## Results

Averaged over the 5 folds

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Accuracy</th>
<th>FPR</th>
<th>FNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotations-Co-Occurrence</td>
<td>77.8</td>
<td>40.0</td>
<td>4.5</td>
</tr>
<tr>
<td>Annotations-LLL05-Patterns</td>
<td>82.3</td>
<td>25.0</td>
<td>10.5</td>
</tr>
<tr>
<td>Words-NaiveBayes</td>
<td>51.3</td>
<td>25.0</td>
<td>95.0</td>
</tr>
<tr>
<td>Words-SVM</td>
<td>73.8</td>
<td>29.0</td>
<td>23.5</td>
</tr>
<tr>
<td>$\phi$-SSLEA</td>
<td>59.8</td>
<td>44.0</td>
<td>33.5</td>
</tr>
<tr>
<td>$C$</td>
<td>73.7</td>
<td>23.5</td>
<td>22.5</td>
</tr>
</tbody>
</table>
Results (II)

- $C$ performs as well as Word-SVM and better than other learning approaches

- accuracies of $C$ and Annotations-Co-Occurrence (which exploits domain knowledge of an expert) are very close

  - Pro: $C$ is composed by patterns (regex) readable
  - Con: time to generate $C$ (hours) $\gg$ time to generate other methods (minutes)
    - but $\approx$ time taken for classifying (seconds)
Conclusions

We proposed:

➔ a method for the automatic synthesis of a classifier for natural language sentences
  ◆ based on syntactic pattern
  ◆ by mean of GP
  ◆ separate-and-conquer

➔ results are highly promising