GpKex: Genetically Programmed Keyphrase Extraction from Croatian Texts

Marko Bekavac and Jan Šnajder

University of Zagreb
Faculty of Electrical Engineering and Computing
Text Analysis and Knowledge Engineering Lab

The Biennial International Workshop on Balto-Slavic Natural Language Processing
Sofia, August 8, 2013
What and why?

- Keyphrases are an effective way to summarize documents
  - *economic crisis, Greece debt crisis, foreign policy, G8 summit*
- Useful for text categorization, document management, search
- Two approaches:
  - keyphrase assignment: keyphrases chosen from a predefined taxonomy
  - keyphrase extraction: keyphrases chosen from document
- Manual keyphrase extraction is tedious and inconsistent
- Many supervised and unsupervised machine learning techniques have been proposed
- We focus on **supervised keyphrase extraction for Croatian** using genetic programming
Genetic programming (GP)

- Evolutionary optimization technique in which solutions are symbolic expressions represented as syntax trees (Koza and Poli, 1992)

**GP in a nutshell**

1. Start with a random set of initial expressions (*population*).
2. Evaluate the *fitness* of each expression from the population.
3. Randomly *select* two expressions, so that best-fitted expressions have a higher chance of being selected.
4. Cross-over selected expressions and replace them with the cross-over result.
5. Occasionally, *mutate* some expressions by changing them slightly.
6. Repeat from step (1) until population fitness converges.
Typically done in two steps:

- **Step 1:** Candidate extraction
  - E.g.: *economic crisis* vs. *crisis in*
- **Step 2:** Candidate scoring using a **keyphrase scoring measure** (KSM)
  - E.g.: *economic crisis* vs. *recent crisis*

Previous approaches learn KSMs using decision trees (Turney, 1999), naïve Bayes (Witten et al., 1999), and SVM (Zhang et al., 2006)

Work for Croatian: naïve Bayes (Ahel et al., 2009), tf-idf scoring (Mijić et al., 2010), topic clustering (Saratlija et al., 2011)

Unlike previous work, we learn KSMs using GP

GP yields interpretable and efficient KSMs
Step 1: Candidate extraction

- Any sequence of words that
  - does not span over clause boundaries
  - matches any of the predefined POS patterns

- Each candidate is assigned a set of features
  - **Frequency-based:**
    - relative term frequency, idf, tf-idf
  - **Position-based:**
    - first/last occurrence, occurrence in title,
    - # occurrences in 1st/2nd/3rd third
  - **Surface form:**
    - length, # discriminative words
Step 2: Genetic programming

- Each genetic expression is a KSM represented as a syntax tree
- Outer nodes: keyphrase features
- Inner nodes: $+, -, \times, /, \log \cdot, \cdot \times 10, \cdot /10, 1/\cdot$
GP parameters

- **Fitness**: Evaluated by comparing top $k$-ranked extracted phrases against gold-standard keyphrases
- **Parsimony pressure**: To prevent overfitting, we use a regularized fitness function:
  \[ f_{\text{reg}} = \frac{f}{1 + N/\alpha} \]
- **Crossover**: Exchanges subtrees rooted at random nodes
- **Mutation**: Grows a random subtree rooted at a randomly chosen node
- **Selection**: Fitness-proportionate with elitist strategy
- **Population**: 500 expressions, maximum 50 generations
Evaluation – Dataset

- 1020 newspaper documents annotated by professional documentalists (Mijić et al., 2010)
- Split into:
  - 960 training docs, each annotated by a single annotator
  - 60 testing docs, each independently annotated by eight annotators
- We use the training set to define a set of six POS patterns: N, AN, NN, NSN, V, X
  - cover ~70% of keyphrases, reduce candidates by ~80%
  - keyphrases of at most length 3 (~93%)
Keyphrase extraction is a highly subjective task
- average human performance: $\sim 65\%$ F1 (Saratlija et al., 2011)

We aggregate human annotations to obtain a ranked list of keyphrases for each document

Evaluation measures:
- Generalized average precision (GAP) (Kishida, 2005)
- $P@10$ and $R@10$ at two agreement levels:
  weak (2-annotator agreement) and strong (5-annotator agreement)
## Results

<table>
<thead>
<tr>
<th>Model</th>
<th>GAP</th>
<th>Strong agreement</th>
<th>Weak agreement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P@10</td>
<td>R@10</td>
<td>P@10</td>
</tr>
<tr>
<td>No parsimony</td>
<td>13.0</td>
<td>8.3</td>
<td>28.7</td>
</tr>
<tr>
<td>$\alpha = 1000$</td>
<td>12.8</td>
<td>8.2</td>
<td>30.2</td>
</tr>
<tr>
<td>$\alpha = 100$</td>
<td>12.5</td>
<td>7.7</td>
<td>27.3</td>
</tr>
<tr>
<td>All POS patterns</td>
<td>9.9</td>
<td>5.1</td>
<td>25.9</td>
</tr>
<tr>
<td>Baseline: tf-idf</td>
<td>7.4</td>
<td>5.8</td>
<td>22.3</td>
</tr>
<tr>
<td>Saratlija et al. (2011)</td>
<td>6.0</td>
<td>5.8</td>
<td>32.6</td>
</tr>
</tbody>
</table>

- First two models perform best and outperform the baseline (except for weak R@10)
- Parsimony pressure does not help, conservative POS filtering does
- Outperforms unsupervised extraction on GAP and strong F1@10
Tf-idf, First, and Rare positively correlated with keyphraseness
Length negatively correlated with keyphraseness
**Summary**

- **Gpkex** uses genetically programmed keyphrase extraction measures to assign ranking to keyphrase candidates.
- Performs comparable to other machine learning methods developed for Croatian ⇒ efficient alternative to more complex models.
- We use simple features ⇒ easily applicable to other languages.
- Data/source code available from [takelab.fer.hr/gpkex](http://takelab.fer.hr/gpkex).
- Future work:
  - use additional (e.g., syntactic) features.
  - learn keyphrase ranking directly.


