A Similarity Measure for Formal Languages Based on Convergent Geometric Series

University of Kassel, Germany

- Florian Bruse
- Maurice Herwig
- Martin Lange

26th International Conference on Implementation and Application of Automata (CIAA'22).

Rouen, June 28-July 1 2022

Motivation

- **Distance measure between regular languages**
- For automatic feedback / grading on the solutions of students.
  - Evaluation of the semantic error instead of the syntactic error.
- Other fields of application: image processing, bioinformatics, data science, etc.

Problems with other distance measures

Word distance induces a distance on languages

$$\hat{d}\,(L_1, L_2) := \min\{d(w_1, w_2) \mid w_i \in L_i\}$$

- For typical distance types d: Euclidian, Manhattan, Cosine, Hamming, Levenshtein, etc.
- Ignores the inner structure of these languages.

Hausdorff distance

$$\hat{d}\,(L_1, L_2) = \max\left(\{\tilde{d}\,(L_1, w) \mid w \in L_2\} \cup \{\tilde{d}\,(L_2, w') \mid w' \in L_1\}\right)$$
$$\text{where } \tilde{d}\,(L, w) = \min\{d(w', w) \mid w' \in L\}$$

- Undecidability problems by using this definition ([Choffrut and Pighizzini, 2002]).

Weighting of the symmetrical difference $L_1 \triangle L_2$

- $L_1 \triangle L_2$ can be infinite.
- Limit value of the fraction of words per word length does not have to exist ([Alur et al., 2013]).

Solution

- Word lengths, descending weight by using the geometric series.

Weight of a Language

Let $L \subseteq \Sigma^*$ and $L^n = \{w \mid w \in L \text{ and } |w| = n\}$.
Let $f_L\colon \mathbb{N} \to [0,1]$ be the fraction of words from Σ^n that are in L.

$$f_L(n) = \frac{|L^n|}{|\Sigma^n|}$$

The $\lambda \in (0,1)$ weight of a language $\omega_\lambda\colon 2^{\Sigma^*} \to [0,1]$ can be determined as follows.

$$\omega_\lambda(L) = (1-\lambda) \cdot \sum_{i=0}^{\infty} \lambda^i \cdot f_L(i) = (1-\lambda) \cdot \sum_{i=0}^{\infty} \lambda^i \cdot \frac{|L^i|}{|\Sigma^i|}$$

$\Rightarrow$ For all $\lambda \in (0,1)$: ω_λ is monotonic, $\omega_\lambda(\emptyset) = 0$ and $\omega_\lambda(\Sigma^*) = 1$.

$L_j = a\Sigma^j$ with $\Sigma = \{a, b\}$ and $\lambda = 0.5$

In [4]:

```
interactive(
    fractions,
    n = IntSlider(value= 2, min= 1, max= 10, layout=Layout(width='500px'), description=r'\(\ j\)'),
    button = ToggleButtons(options=['lin-scale', 'log-scale'], description=" "),
)
```

For a DFA the weight is computable

Theorem. $\omega_\lambda(L(\mathcal{A}))$ is computable if $\mathcal{A} = (Q, \Sigma, \delta, q_i, Q_F)$ is a DFA.

Proof-idea:

For $q \in Q$, let $L_q = L(\mathcal{A}_q)$ with $\mathcal{A}_q = (Q, \Sigma, \delta, q, Q_F)$ and the weight.

$$\omega_\lambda(L_q) = (1-\lambda) \cdot \sum_{i=0}^{\infty} \lambda^i \cdot \frac{|L_q^i|}{|\Sigma^i|}$$

By using the transitions, the weights of the languages can be determined recursively.

$$(1-\lambda) \cdot |L_q^0| + \frac{\lambda}{|\Sigma|} \cdot \sum_{a \in \Sigma} \omega_\lambda(L_{\delta(q,a)})$$

- $t_{q,q'} = \frac{|\{a \in \Sigma | \delta(q,a)=q'\}|}{|\Sigma|}$
- $e_q = 1$ if $q \in Q_F$ and $e_q = 0$ otherwise

So we can write $\omega_\lambda(L_q)$ as:

$$\omega_\lambda(L_q) = (1-\lambda) \cdot e_q + \lambda \cdot t_{q,q_1} \omega_\lambda(L_{q_1}) + \cdots + \lambda \cdot t_{q,q_n} \omega_\lambda(L_{q_n})$$

This results in an equation system:

$$\begin{array}{rcl} -(1-\lambda) \cdot e_{q_1} &=& (\lambda \cdot t_{q_1,q_1} - 1) \cdot \omega_\lambda(L_{q_1}) + \cdots + \lambda \cdot t_{q_1,q_n} \cdot \omega_\lambda(L_{q_n}) \\ \vdots & & \vdots \qquad\qquad \ddots \qquad\qquad \vdots \\ -(1-\lambda) \cdot e_{q_n} &=& \lambda \cdot t_{q_n,q_1} \cdot \omega_\lambda(L_{q_1}) + \cdots + (\lambda \cdot t_{q_n,q_n} - 1) \cdot \omega_\lambda(L_{q_n}) \end{array}$$

There is exactly a unique solution for the system of equations and:

$$\omega_\lambda(L) = \omega_\lambda(L_{q_i})$$

$\Rightarrow \omega_\lambda(L(\mathcal{A}))$ is computable in time $\mathcal{O}(n^3)$ for a DFA with n states.

Distance of two Languages

Aim: weight function $\to$ distance function.
Distance of two languages $d_\lambda\colon 2^{\Sigma^*} \times 2^{\Sigma^*} \to [0, 1]$
can be determined by the symmetric difference $L_1 \triangle L_2$.

$$d_\lambda(L_1, L_2) = \omega_\lambda(L_1 \triangle L_2)$$

Theorem. d_λ is a metric on the space of all Σ-languages.

Practical application

students evaluation

Implementation: **https://github.com/maurice-herwig/wofa**

Practical application

Recap motivation: automatic feedback / grading on the solutions of the students.

- Standard exercise: construct an automaton that recognizes exactly a given language.
- Automatic feedback initiates an error-driven learning cycle.

Good parameter values?

- For feedback / grading to the students solution.
- For this we consider the difference of the following **example** languages over $\Sigma = \{a, b\}$.

$$L_{target} = \{w \in \Sigma^* \mid w \text{ contains the subword } ab\}$$
$$L_{submission} = \{ab\}$$

$L_{target} = \{w \in \Sigma^* \mid w \text{ contains the subword } ab\}$ and $L_{submission} = \{ab\}$

In [5]:

```
interactive(
    weight_calc_lam,
    lam = FloatSlider(value=0.5, min=0.1, max=0.9, layout=Layout(width='500px'), description=r'\(\lambda\)'),
)
```

Redistribution of Weights on Short Words

Problem λ

ω_λ results in a strong overweighting of short word lengths. - η near to 1 flatten the distribution curve, but this is not a got distribution over the length. - Does not lead to good results.

Solution: parameter η

- Words with length up to a certain length η should be weighted equally.
- Word with length greater than η should be weighted exponentially decreasing.

$$\omega_\lambda^\eta(L) = \omega'_\lambda(L^{(\leq\eta)}) + \omega_\lambda(L^{(>\eta)})$$

$$\omega_\lambda^\eta(L) = \omega_\lambda'(L^{(\leq\eta)}) + \omega_\lambda(L^{(>\eta)})$$

Theorem. $\omega_\lambda^\eta(L)$ is computable.

Proof-idea:

- $\omega_\lambda'(L^{(\leq\eta)})$ by matrix multiplication.
  - A^1 = Adjacency matrix of δ.
  - $A^i = A^{i-1} \cdot A^1$
  - Weight of one word $\frac{1-\lambda^{\eta+1}}{\sum_{i=0}^{\eta} |\Sigma|^i}$.
- $\omega_\lambda(L^{(>\eta)}) = \omega_\lambda(L) - \omega_\lambda(L^{(\leq\eta)})$

$\Rightarrow \omega_\lambda'(L^{(\leq\eta)})$ is computable in time $\mathcal{O}(\eta \cdot |Q|^3)$.

In [6]:

```
interactive(
    weight_calc_eta_and_lam,
    lam = FloatSlider(value=0.7, min=0.1, max=0.9, layout=Layout(width='500px'), description=r'\(\lambda\)'),
    eta = IntSlider(value= 4, min= 0, max= 10, layout=Layout(width='500px'), description=r'\(\eta\)')
)
```

Good parameter values: η = pumping constant, λ so that $\omega'_\lambda(\Sigma^{*^{(\leq\eta)}}) = \omega_\lambda(\Sigma^{*^{(>\eta)}})$

Summary

- Weight function for regular languages.
  - Well defined by using the geometric series.
- Distance measure between two regular languages.
  - By the determinations of the weight of the symmetrical difference.
- Practical use for students evaluation.
  - Initial part with constant weighting of words.

Currently work

- Web page for automatic correction of student submissions by the distance measure.

Open work

- Is it possible to calculate the distance between two NFA without explicit determination?