Eredmények és ötletek

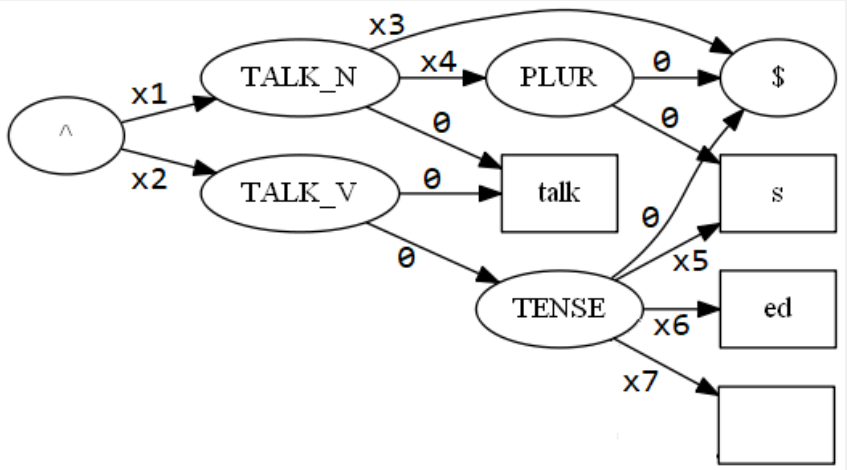
Véges állapotú súlyozott automaták tanulásáról

Borbély Gábor

2019.06.14.

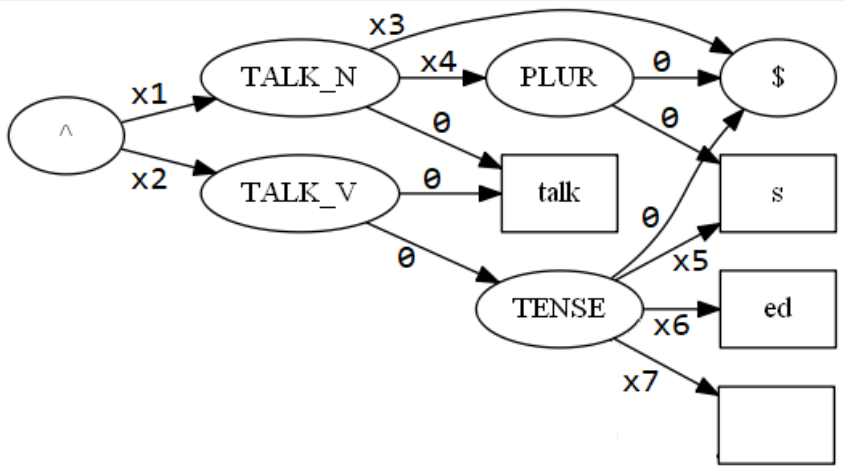
[W]FSA

(wighted) finite state automaton



[W]FSA

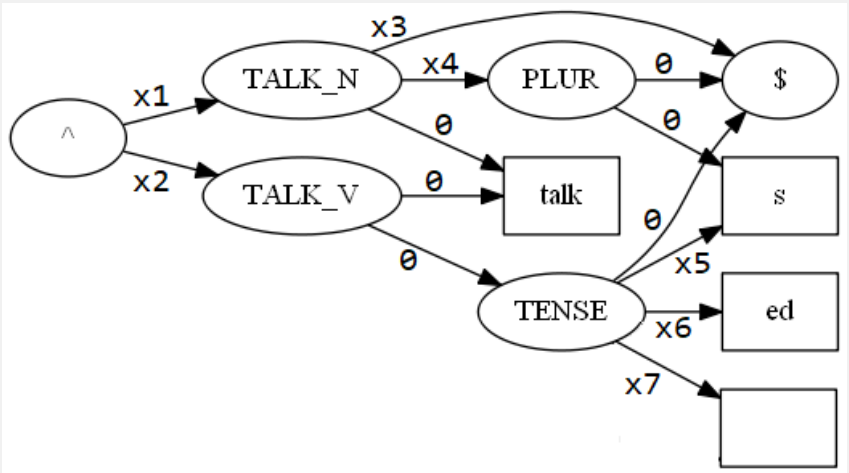
(wighted) finite state automaton



word	paths
talk	^→TALK_V[talk] →TENSE[] →$ ^→TALK_N[talk] →$
talks	^→TALK_V[talk] →TENSE[s] →$ ^→TALK_N[talk] →PLUR[s] →$
talked	^→TALK_V[talk] →TENSE[ed] →$

Struktúra mátrixok

$$C = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \quad M = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$



Struktúra mátrixok

$$C \qquad M \qquad P$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\underline{q}(\underline{x}) = M \cdot \exp(P \cdot \underline{x})$$

$$\underline{g}(\underline{x}) = C^{\top} \exp(\underline{x}) - \underline{1} = \underline{0}$$

Formális felírás

- C megkötések
- M melyik szót hány útvonal generálja
- P melyik útvonalon mely változók vannak

Formális felírás

- C megkötések
- M melyik szót hány útvonal generálja
- P melyik útvonalon mely változók vannak, és hányszor!

Formális felírás

- C megkötések
- M melyik szót hány útvonal generálja
- P melyik útvonalon mely változók vannak, és hányszor!
- M és P meghatározásához egy ND-Recognize algoritmus kell
  - C-hez csak az automata bejárása kell

Formális felírás

- C megkötések
- M melyik szót hány útvonal generálja
- P melyik útvonalon mely változók vannak, és hányszor!
- M és P meghatározásához egy ND-Recognize algoritmus kell
  - C-hez csak az automata bejárása kell
-

$$\underline{x} = \arg\min_{\underline{g}(\underline{x})=\underline{0}} \sum_i p_i \ln \frac{p_i}{q_i(\underline{x})}$$

- Lagrange multiplikátor módszer:

$$F(\underline{x}, \underline{\lambda}) = f(\underline{x}) + \underline{\lambda} \cdot \underline{g}(\underline{x})$$

$$\frac{\partial F}{\partial \underline{x}} = 0, \frac{\partial F}{\partial \underline{\lambda}} = 0$$

- Lagrange multiplikátor módszer Newton módszerrel megoldva

$$\begin{pmatrix} \underline{x} \\ \underline{\lambda} \end{pmatrix}_{\text{next}} = \begin{pmatrix} \underline{x} \\ \underline{\lambda} \end{pmatrix} - \underline{\mathrm{H}}_F(\underline{x}, \underline{\lambda})^{-1} \cdot \mathrm{grad}_F(\underline{x}, \underline{\lambda})$$

- Lagrange multiplikátor módszer Newton módszerrel megoldva

$$\begin{pmatrix} \underline{x} \\ \underline{\lambda} \end{pmatrix}_{\text{next}} = \begin{pmatrix} \underline{x} \\ \underline{\lambda} \end{pmatrix} - \underline{\mathrm{H}}_F(\underline{x}, \underline{\lambda})^{-1} \cdot \mathrm{grad}_F(\underline{x}, \underline{\lambda})$$

-

$$\mathrm{grad}_F(\underline{x}, \underline{\lambda}) = \begin{pmatrix} \mathrm{grad}_f(\underline{x}) + \mathrm{J}_g(\underline{x}) \cdot \underline{\lambda} \\ \underline{g}(\underline{x}) \end{pmatrix}$$

$$\mathrm{H}_F(\underline{x}, \underline{\lambda}) = \begin{pmatrix} \mathrm{H}_f(\underline{x}) + \sum_{j=1}^{k} \lambda_j \cdot \mathrm{H}_{g_j}(\underline{x}) & \mathrm{J}_{\underline{g}}(\underline{x}) \\ \mathrm{J}_{\underline{g}}^{\top}(\underline{x}) & 0 \end{pmatrix}$$

- Lagrange multiplikátor módszer Newton módszerrel megoldva

$$\begin{pmatrix} \underline{x} \\ \underline{\lambda} \end{pmatrix}_{\text{next}} = \begin{pmatrix} \underline{x} \\ \underline{\lambda} \end{pmatrix} - \underline{\mathrm{H}}_F(\underline{x}, \underline{\lambda})^{-1} \cdot \mathrm{grad}_F(\underline{x}, \underline{\lambda})$$

-

$$\mathrm{grad}_F(\underline{x}, \underline{\lambda}) = \begin{pmatrix} \mathrm{grad}_f(\underline{x}) + \mathrm{J}_g(\underline{x}) \cdot \underline{\lambda} \\ \underline{g}(\underline{x}) \end{pmatrix}$$

$$\mathrm{H}_F(\underline{x}, \underline{\lambda}) = \begin{pmatrix} \mathrm{H}_f(\underline{x}) + \sum_{j=1}^{k} \lambda_j \cdot \mathrm{H}_{g_j}(\underline{x}) & \mathrm{J}_{\underline{g}}(\underline{x}) \\ \mathrm{J}_{\underline{g}}^{\top}(\underline{x}) & 0 \end{pmatrix}$$

- Miért?

- Lagrange multiplikátor módszer Newton módszerrel megoldva

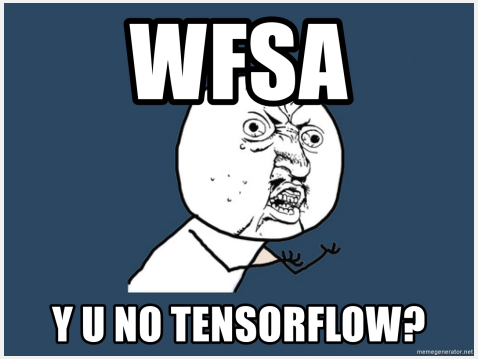
$$\begin{pmatrix} \underline{x} \\ \underline{\lambda} \end{pmatrix}_{\text{next}} = \begin{pmatrix} \underline{x} \\ \underline{\lambda} \end{pmatrix} - \mathrm{H}_F(\underline{x}, \underline{\lambda})^{-1} \cdot \mathrm{grad}_F(\underline{x}, \underline{\lambda})$$

-

$$\mathrm{grad}_F(\underline{x}, \underline{\lambda}) = \begin{pmatrix} \mathrm{grad}_f(\underline{x}) + \mathrm{J}_g(\underline{x}) \cdot \underline{\lambda} \\ \underline{g}(\underline{x}) \end{pmatrix}$$

$$\mathrm{H}_F(\underline{x}, \underline{\lambda}) = \begin{pmatrix} \mathrm{H}_f(\underline{x}) + \sum_{j=1}^{k} \lambda_j \cdot \mathrm{H}_{g_j}(\underline{x}) & \mathrm{J}_{\underline{g}}(\underline{x}) \\ \mathrm{J}_{\underline{g}}^{\top}(\underline{x}) & 0 \end{pmatrix}$$

- Miért?



Implementálás részletek

- Az ND-Recognize elég sok idő, C-ben kell

Implementálás részletek

- Az ND-Recognize elég sok idő, C-ben kell
  - nagyban párhuzamosítható!

Implementálás részletek

- Az ND-Recognize elég sok idő, C-ben kell
  - nagyban párhuzamosítható!
- A Lagrange multiplikátorok miatt nyeregpont, első deriváltat használó módszerek nem jók

Implementálás részletek

- Az ND-Recognize elég sok idő, C-ben kell
  - nagyban párhuzamosítható!
- A Lagrange multiplikátorok miatt nyeregpont, első deriváltat használó módszerek nem jók
- Később kelleni fog a Hesse mátrix másra is!

Implementálás részletek

- Az ND-Recognize elég sok idő, C-ben kell
  - nagyban párhuzamosítható!
- A Lagrange multiplikátorok miatt nyeregpont, első deriváltat használó módszerek nem jók
- Később kelleni fog a Hesse mátrix másra is!
- A Hesse mátrix ritka, Intel MKL

Implementálás részletek

- Az ND-Recognize elég sok idő, C-ben kell
  - nagyban párhuzamosítható!
- A Lagrange multiplikátorok miatt nyeregpont, első deriváltat használó módszerek nem jók
- Később kelleni fog a Hesse mátrix másra is!
- A Hesse mátrix ritka, Intel MKL

Implementálás részletek

- Az ND-Recognize elég sok idő, C-ben kell
  - nagyban párhuzamosítható!
- A Lagrange multiplikátorok miatt nyeregpont, első deriváltat használó módszerek nem jók
- Később kelleni fog a Hesse mátrix másra is!
- A Hesse mátrix ritka, Intel MKL
  DL4MT sztori

Implementálás részletek

- Az ND-Recognize elég sok idő, C-ben kell
  - nagyban párhuzamosítható!
- A Lagrange multiplikátorok miatt nyeregpont, első deriváltat használó módszerek nem jók
- Később kelleni fog a Hesse mátrix másra is!
- A Hesse mátrix ritka, Intel MKL
  DL4MT sztori
  kiegyszerűsödés

Automaták összehasonlítása

- Borbély, Kornai *Sentence Length*, to appear in: MOL2019

$$
\begin{aligned}
&-\lambda \cdot \ln \lambda + \sum_{x \in X \cap \mathrm{supp}(\mathcal{H}_i)} p_x \cdot \ln \frac{p_x}{\mathbb{Q}_{\mathbf{w}_i^*}(x)} + \\
&\frac{1}{|D|} \cdot (\ln \mathrm{Vol}(\mathcal{H}_i) + \ln \mathrm{Vol}(\text{aux. model})) + \\
&\frac{1}{2|D|} \cdot \ln \det (\text{model Hessian}) + \\
&\frac{1}{2|D|} \cdot \ln \det (\text{aux. model Hessian}) + \\
&\frac{d'}{2|D|} \cdot \ln \frac{|D|}{2\pi}
\end{aligned}
$$

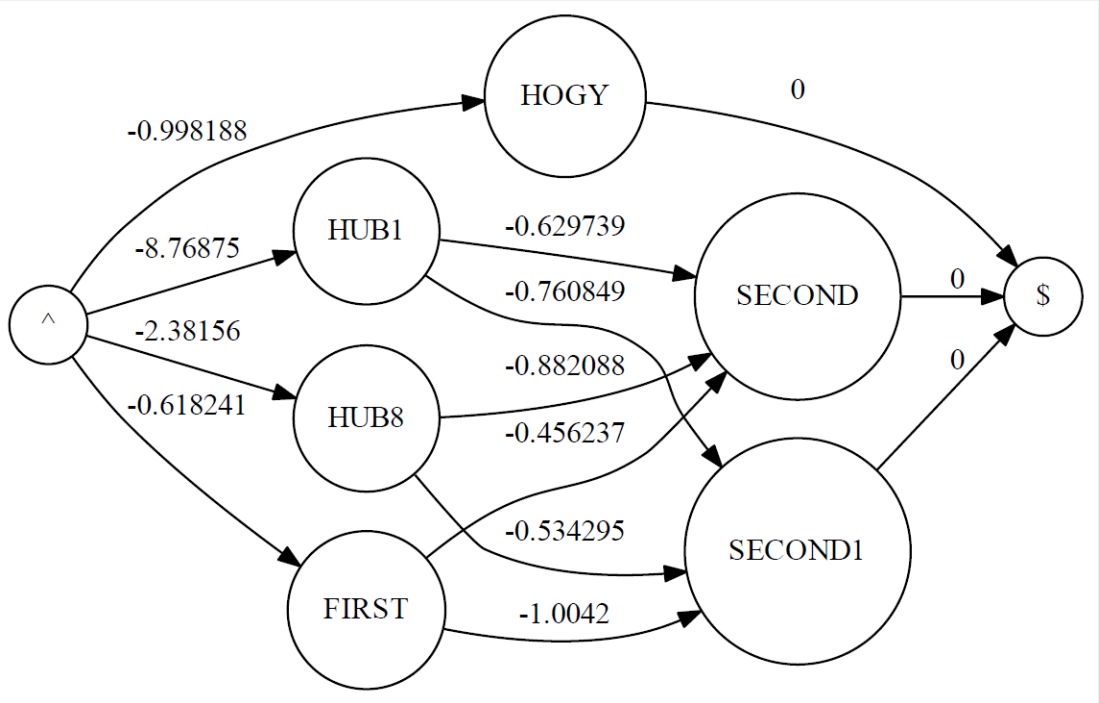
ahol

- d' a modell paraméterek és a kiegészítő paraméterek számának összege
- λ a közös tartó mértéke

Hibrid optimalizálás

- adott FSA-ra a súlyokat meg lehet találni
- két FSA között az előző képlet dönt
- az FSA-k között genetikus algoritmussal keresünk
  - Csikós Dominik: *Véges állapotú automaták tanulása genetikus algoritmussal*
  - `http://math.bme.hu/~csikosd/Szakdolgozat.pdf`
  - `https://github.com/csikosd`
- Mutációk: összevonás és szétdobás
- minden populációban a legjobb 20 megtartása

Hibrid optimalizálás – eredmények



^	” ”
FIRST	„”, a, minden
HUB1 HUB8	egyvala, másvala se, más, akár, vala, bár
HOGY	hogy
SECOND SECOND1	hogy, honnan, hovA, ki, meddig, mely, melyik, miErt, mikor hAny, hol, honnEt, hova, kor, mennyi, mi, milyen

További irányok

További irányok

- ND-Recognize gyorsítása

További irányok

- ND-Recognize gyorsítása
  - → GPU-n

További irányok

- ND-Recognize gyorsítása
  - → GPU-n
- A WFSA-tanulás analízise

További irányok

- ND-Recognize gyorsítása
  - → GPU-n
- A WFSA-tanulás analízise
  - → quasi-Newton

További irányok

- ND-Recognize gyorsítása
  - → GPU-n
- A WFSA-tanulás analízise
  - → quasi-Newton
- „constrained Hessian”

További irányok

- ND-Recognize gyorsítása
  - $\rightarrow$ GPU-n
- A WFSA-tanulás analízise
  - $\rightarrow$ quasi-Newton
- „constrained Hessian”
  - $\rightarrow$ $LDL^\top$ dekompozíció

További irányok

- ND-Recognize gyorsítása
  - $\rightarrow$ GPU-n
- A WFSA-tanulás analízise
  - $\rightarrow$ quasi-Newton
- „constrained Hessian”
  - $\rightarrow$ $LDL^\top$ dekompozíció
- az FSA-k kifejező ereje általában, általánosítani kéne

További irányok

- ND-Recognize gyorsítása
  - → GPU-n
- A WFSA-tanulás analízise
  - → quasi-Newton
- „constrained Hessian”
  - → $LDL^\top$ dekompozíció
- az FSA-k kifejező ereje általában, általánosítani kéne
  - → [W]CFSA: Compositional Finite State Automaton