`emtsv` – Egy formátum mind felett

Indig Balázs, Sass Bálint, Simon Eszter,
Mittelholcz Iván, Kundráth Péter, Vadász Noémi

MTA Nyelvtudományi Intézet
ELTE Bölcsészettudományi Kar

2019. január 25.
MSZNY 2019, Szeged

motiváció

kiindulópont: eredeti `e-magyar`
= GATE keretrendszer által integrált
magyar nyelvfeldolgozó modulok

A felhasználói visszajelzések alapján
a fő problémás elem a GATE volt mint integráló keretrendszer.

gond:

- a GATE nem kikerülhető
- a GATE nehézkes: bonyolult installálás, nagy overhead, nehéz hibakeresés
- a GATE által adott kommunikációs formátum kényelmetlen

motiváció

a GATE által adott kommunikációs formátum:

GATE XML

- speciális *standoff* annotáció
- belső formátum
  $\rightarrow$ további, GATE-en kívüli felhasználásra nem kényelmes
- a standoff annotáció lehetetlenné teszi
  az adatfolyamként való feldolgozást
  $\rightarrow$ nagy fájlok feldolgozására nem alkalmas
- két lépés közötti köztes formátumnak szintén azért nem jó,
  mert nem dolgozható fel adatfolyamként

cél

- a GATE kiváltása
- használhatóság növelése, egyszerűség
- alkalmas kommunikációs formátum
- a modulok önálló *és* láncba épített használatának egyenrangú támogatása

terv

a GATE XML helyett találjunk ki egy egyszerű *adatformátumot*,
és (a GATE helyett) hozzunk létre egy azt kezelő *keretrendszert*,
illesszük bele az eredeti `e-magyar` modulokat

= cseréljük le a GATE-et mint integráló eszközt

ha már változtatunk, nézzünk távolabbra

1. loosely coupled rendszert szeretnénk

könnyen alkalmazható → *könnyen továbbfejleszthető*
(= külső fejlesztők saját moduljaikat könnyen a rendszerhez tudják illeszteni)

az alternatív rendszerek nem ilyenek / kevésbé ilyenek

A magyarlánc esetében nehézkes a rendszer módosítása,
új modulokkal való bővítése, a modulok *lecserélése, összehasonlítása*.

Bizonyos mértékben igaz ez
a nem specifikusan magyar megoldásokra is: UDPipe és spaCy.

(*kitérő:* az `emtsv` modulok sorában fontos elem a jó minőségű,
speciálisan magyar nyelvre kialakított morfológiai elemző.)

ha már változtatunk, nézzünk távolabbra

2. webes API-t szeretnénk

lehetőséget a felhőben való üzemeltethetőségre

Az ilyennel bíró rendszerek (pl.: WebSty, Weblicht) esetében
a szoftver forráskódja nem érhető el saját példány futtatása céljából,
külső fejlesztőként létrehozott modulok beillesztése a láncba
nem lehetséges.

ha már változtatunk, nézzünk távolabbra

2. webes API-t szeretnénk

lehetőséget a felhőben való üzemeltethetőségre

Az ilyennel bíró rendszerek (pl.: WebSty, Weblicht) esetében
a szoftver forráskódja nem érhető el saját példány futtatása céljából,
külső fejlesztőként létrehozott modulok beillesztése a láncba
nem lehetséges.

tehát a GATE kiváltásán túl 2 dolgot szeretnénk egyszerre:
1. loosely coupled rendszer
2. webes API

loosely coupled = nyíltan integrált

– független („egymásról mit sem tudó”) alkotóelemek
– egyszerűen, könnyen helyettesíthető modulok
– vminek a módosításához semmi mást nem kell módosítani
  ← jól definiált/elhatárolt egységek
– könnyebb változtatás/módosíthatóság,
  könnyebb továbbfejlesztés/kiterjeszthetőség – nyitottság
– fontos az elemek közötti (szabványos) kommunikáció/koordináció
  ← a *standard adatformátum* elősegíti

– *elv:* érdemes a couplingot csökkenteni – a szabványosítás javára
  → ezt csináljuk!

a) egységes adatformátum: fejléces tsv

```
form     lemma xpostag
A        a     [/Det|Art.Def]
kutyák   kutya [/N][Pl][Nom]
ugatnak  ugat  [/V][Prs.NDef.3Pl]
.        .     [Punct]

A        a     [/Det|Art.Def]
...
```

egy token – egy sor
oszlopok elválasztása egy TAB karakterrel
mondat végén üres sor
fejléc!

lényegében kompatibilis a CONLL-U formátummal

a) egységes adatformátum: fejléces tsv

– egyszerű formátum; könnyű, szabványos feldolgozhatóság

– adatfolyamként feldolgozható – nagyon fontos
(a feldolgozás során nem kell az egész fájlt a memóriában tartani)

– adat: szabad szöveg vagy JSON (újsor és TAB nélkül)

– egymástól elkülönülő, egymást nem zavaró annotációk (oszlopok)

– *vezérlés:* fejléc – ez határozza meg a rendszer működését

→ a *formátum révén* valósulnak meg a kitűzött céljaink!

b) integráló architektúra: `xtsv` keretrendszer

újonnan fejlesztett, általános célú keretrendszer
a tsv általi kommunikációt valósítja meg általánosan
ez fogja össze a loosely coupled modulokat

vezérlés: **fejléc** – ez határozza meg a rendszer működését

minden modul

1. a fejlécben definiált *oszlopnevek* segítségével azonosítja a szükséges bemeneti adatai helyét (az oszlopok sorrendjétől függetlenül);
2. kimeneti adatait szintén a fejlécben definiált nevű új oszlopokba helyezi el (mindig a meglévő oszlopok után, a végére);
3. az összes többi oszlopot változatlanul hagyja

Elengedhetetlen, hogy az oszlopok elnevezése és tartalma
az egymásra épülő modulok között szinkronizálva legyen.

b) integráló architektúra: `xtsv` keretrendszer

Technikailag minden modulnak két függvényt kell definiálnia:

- `prepare_fields()`
  definiálja az input oszlopok nevét, feldolgozásának módját
- `process_sentence()`
  implementálja magát a feldolgozó lépést, melyben a kívánt számú mezőt illesztjük az egyes tokenek „végére”

A modulok egy config fájlban vannak deklarálva néhány jellemzővel. Ha új modult szeretnénk a rendszerbe illeszteni, vagy egy modult egy másik modellel szeretnénk használni, néhány sort kell ebben a config fájlban átírni. → könnyű módosíthatóság!

Az `xtsv` dinamikusan *ellenőrzi*, létrehozza és futtatja a kívánt láncot.
A moduloknak csak az `xtsv` fenti követelményeinek kell megfelelniük.

c) `xtsv` – használati módok

Az `xtsv` a rendelkezésre álló modulokból
létrehozza a kívánt láncot, ellenőrzi és futtatja.

Kétféle használati mód szerint.

1. CLI (← paraméterrel)

```
$ cat file | python3 emtsv.py morph,pos
$ cat file | python3 emtsv.py morph | python3 emtsv.py pos
```

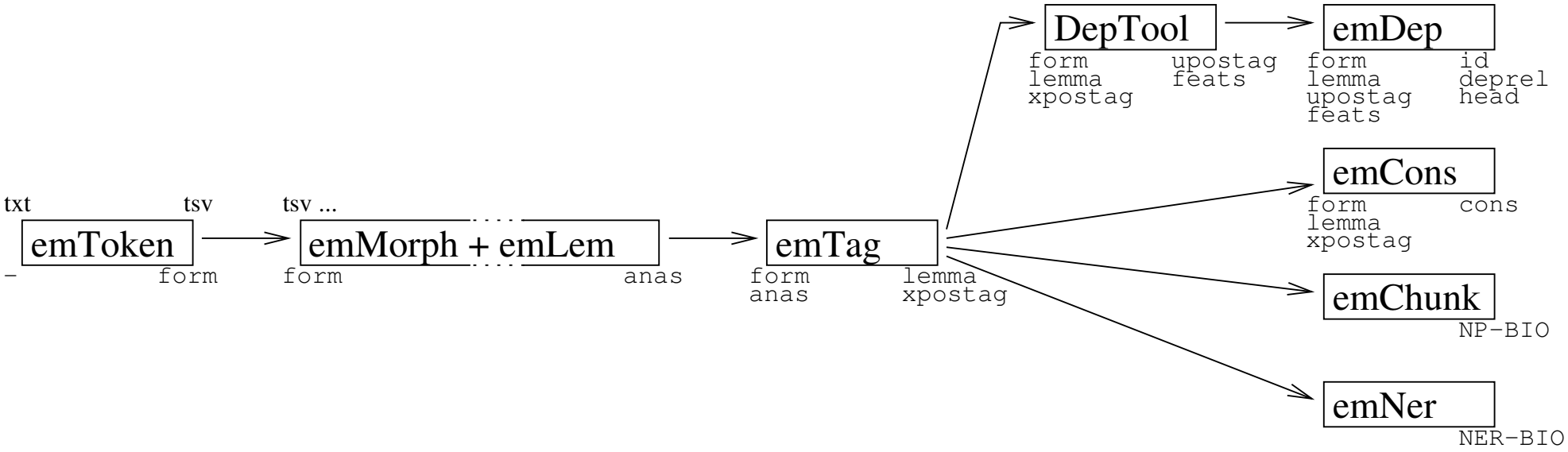
2. REST API (← paraméter nélkül)

A CLI mellett automatikusan REST API-t is kapunk,
futtathatjuk saját felhőben.

```
$ python3 emtsv.py
http://127.0.0.1:5000/morph/pos
```

← itt érhető el, az input adat HTTP POST kéréssel küldendő

modulok



modulok az `xtsv`-hez illesztve:

`xtsv` + `e-magyar` modulok = `emtsv`

a modulok funkcionalitása megmaradt

Java nyelvű modulok → Python modulba csomagolva

ad hoc adatformátumok helyett: JSON illetve natív adatszerkezetek

a modulokról külön-külön

- `emToken` – átdolgozás, stdin/stdout kommunikáció
- `emMorph` + `emLem` – utóbbiból Python változat
- `emTag` – a PurePOS hagyományos formátumához képest az új tsv+JSON formátum fontos előrelépés
- `emChunk` és `emNER` – oszlopnév-alapú működés; az átdolgozott bemenetiformátum-kezelés szolgált az `xtsv` alapjául
- `emMorph2UD` *(új!)* – az `emMorph` által kiadott morfoszintaktikai információkat konvertálja át az Universal Dependencies-nek megfelelő jegy–érték párok sorára (ld. külön poszter)
- `emDep` – különválasztottuk, átállítottuk az UD modell használatára
- `emCons` – különválasztottuk

`emMorph+emLem` saját REST API

`https://emmorph.herokuapp.com/dstem/`*`terem`*

A morfológia böngészőből hívható módon külön rendelkezésre áll, telepítés nélkül, emberi fogyasztásra is alkalmas kimenettel.

REST API = szabványosan, programnyelvtől függetlenül, távolról is hívható


```
{
  "terem": [
    {
      "lemma": "terem",
      "morphana": "terem[/N]=terem+[Nom]=",
      "readable": "terem[/N] + [Nom]",
      "tag": "[/N][Nom]",
      "twolevel": "t:t e:e r:r e:e m:m :[/N] :[Nom]"
    },
    ...
  ]
}
```

következmény: bárhol be lehet lépni a láncba

a nyílt integrációnak és az új adatformátumnak köszönhetően
a modulok önállóan és láncba kötve egyaránt használhatók
$\rightarrow$
a szerelőszalag *tetszőleges szakasza* lefuttatható,
bárhol be lehet lépni a láncba, és ki lehet lépni belőle
$\rightarrow$
a modulok között szabadon rendelkezünk az adattal

pl. kézi módosítás/javítás beillesztése:

```
cat input | python3 emtsv.py tok,morph,pos > interm
edit interm
cat interm | python3 emtsv.py conv-morph,dep > output
```

következmény: könnyű kiterjeszthetőség

új modulok beillesztése egyszerű

- 1. minta: `DummyTagger/dummy.py`
- 2. copy
- 3. `prepare_fields()` – 2-3 sor: input oszlopnevek
- 4. `process_sentence()` – az egy mondatot feldolgozó kód
- 5. `config.py` – 3-4 sor: `source_fields`, `target_fields` (a többi bejegyzés mintájára)

Kész! :)

összefoglalás

Az eredeti `e-magyar` rendszer továbbfejlesztésével,
a GATE mint integráló eszköz kiváltásával
létrehoztunk a magyar nyelvre egy funkciógazdag,
szabadon elérhető elemzőláncot.

- nyíltan integrált (*loosely coupled*), moduláris
  - módosítható, továbbépíthető, új modulokra nyitott
  - tanulmányozható, a modulok lecserélhetők, összehasonlíthatók
- szabványos kommunikációs formátummal bír
- CLI-vel és REST API-val rendelkezik

Elérhető: `https://github.com/dlt-rilmta/emtsv`