Tanulságok magyar mondatellenőrző nyelvi adatainak átvitelénél

Naszódi Mátyás, e-mail: naszodim@morphologic.hu

MorphoLogic, 1122 Ráth György utca 36.

Kivonat A mondatellenőrzők a mondatok nyelvtani helyességét nem képesek eldönteni, csupán hibás eseteket keresnek a mondatban, melyekre nem derül fény a szóellenőrzésből. Jóságát abban mérik, hogy a valós hibák hány százalékét ismeri fel, illetve hányszor riaszt feleslegesen. Az elérhető minőség függ az előállítás módjától. A leíró formalizmusok és meghajtó programok lehetőségei korlátosak. Jelen cikk kitér két eszköz képességeire, és bemutatja, hogyan (nem) lehet portolni az egyik formalizmusból a másikba a nyelvi tudást. A próbálkozás egyéb nyelvészeti tanulságokkal is szolgál. Rámutat morfológiai leírások minőségi jellemzőire és nyelvi jelenségekre. Végül keresi a további fejlesztés irányát.

Kulcsszavak: mondatellenőrzés, statisztika, nyelvtan, nyelvminőség

1. Bevezető

A mondatellenőrzők a szóellenőrzők után nem sokkal, a 80-as években megjelentek. Angol, majd francia, spanyol nyelvekhez készítettek – először a Grammatik, később más műhelyek is – grammar checker-eket. Magyarra a 90-es évek közepén jelent meg, először a MS Office részeként. A késést nyelvünk összetettsége okozta. Nem csak a szótan okoz nehézséget, hanem nyelvtanunk is lényegesen elüt az indoeurópai nyelvek szerkezetétől.

A mondatellenőrzők nem azt ellenőrzik, hogy a leírt mondat megfelel-e a nyelv formálisan megadott nyelvtanának. Formálisan megadott szabályokba belefér olyan mondat, mely szemmel láthatóan (humán olvasatra) helytelen. Az angol nyelvnél talán felismerhető lehet a hibák mérhető hányada ezen a módon, de az angolnak egyrészt kötött szórendű a nyelvtana, másrészt az angol mondatok nehezen tűrik a hiányt. A szabad szórend és az erősen hiányos mondatok megengedése reménytelenné teszik a globális nyelvtan módszerét. Inkább negatív szabályokat adnak meg szép számmal, ahelyett, hogy azt írnák le, mi a helyes.

1.1. Létező programok

Magyar nyelvre három ilyen célú programot ismerek. Az egyik a MorphoLogic 95-ben termékké váló Helyesebb programja[1] - ezt használta korábban a Microsoft az Office-ban, a másik a Németh László és társai által 2009-ben készített Lightproof[2], mely a LibreOffice-ba integrálható, és harmadik az utóbbi időben megjelent Microsoft Office 365-be beépített nyelvhelyesség-ellenőrző. Más nyelvekre több is létezik, de csak kevés üti meg a hasznos szintet.

A programok minőségét nem vetem összehasonlító teszt alá, mint ahogy a helyesírás-ellenőrzőknél szokásos, mert a rangsorban objektív okokból egyértelmű: a Helyesebb a legjobb, ezt követi a Lightproof a nyelvi információ átvétele után is, majd használhatatlanul a sor végén a Microsoft saját fejlesztése.

1.2. Mondatellenőrző mint a szóellenőrző kiterjesztése

Ezeket a programokat egyesek a helyesírás-ellenőrzők kiterjesztésének tekintik. Amikor a hiba szószinten nem fedezhető fel, a szövegkörnyezet rámutathat az elírásra. Különösen a magyar nyelv szempontjából fontos, ahol a nagy szóalakszám miatt a kellő lefedettség érdekében ritkább szóalakok is bekerülnek a helyesnek ítéltek közé. Ráadásul nyelvünk igen sűrű: a szavak közt az átlagos távolság kicsi.[3] Emiatt nem árt az utólagos szövegkörnyezeti ellenőrzés. De sok más célra is felhasználható.

2. Technikai áttekintés

2.1. A hibák lefedettsége

A mondatellenőrző nem a helyességet ellenőrzi, hanem a helytelenséget fedi fel nyelvtani, még gyakrabban statisztikával felderített szabályok alapján. Hogy egy szó helyes-e, arról a helyesírás-ellenőrző is hasonlóan dönt. A helyes szóalakokat morfológiai szabályokkal adjuk meg, de sokszor ezen felül beleszól a szóstatisztika is. Így lehetséges, hogy szabályos szavakat, szóalakokat elutasít.

Míg szószinten nagy lefedettséget kis mellélövéssel el lehet érni, mondatszinten ezek az arányok eltérnek, mert a lehetséges esetek száma több nagyságrenddel nagyobb. Míg szóalakból egy nyelven több milliárd helyes van (nem ragozó nyelveknél ez lehet csupán több százezer), az ezekből alkotott helyes mondatokból ennek exponenciálisszorosa. A várható helytelenek számánál az arány jobban eltér. Ahhoz, hogy a (mondatszintű) hibák 90 százalékát jelezzük, sokkal nagyobb adatbázisra lenne szükség, mint a szóellenőrzőnél, ahol a kilencvenszázalékos követelmény természetes.[4] Ekkora adatbázist előállítani képtelenség. A minőségi követelmény szintjét máshol kell meghúzni. Ha a mondatszintű hibák több mint felét felderíti egy ellenőrző, akkor már hasznos.

Ha tökéletes a szótan adatbázisa, akkor a szóellenőrző helytelen alakokat nem fogad el. Ezt még felülbírálhatja a statisztika. Egy ellenőrző sem tartja helyesnek a képzett *tanit* szót, és több szóellenőrző megugatja az *egyenlőre* alakot. Az első esetben teljesen jogos a döntés. A *tanit* alak egyetlen természetes szövegkörnyezetben sem helyes. Az utóbbi előfordulása az esetek kicsi, de nem elhanyagolható részében kifejezetten megfelelő: *Nem szabták egyenlőre a nadrág két szárát.*

Egy nyelvi ellenőrző jóságát durván két paraméterrel lehet mérni.

- Hibás esetek hány százalékában jelez
- Hibátlan esetek hány százalékában jelez feleslegesen.

Tegyük fel, hogy egy kérdéses szóalak száz előfordulása közül statisztikailag egyszer helyes és kilencvenkilencszer hibás a szövegben, tehát a szó ugyan helyes, de nem a szövegkörnyezetben. Ekkor 1000 előfordulás közül 990 hibás. (Persze egy literátusabb szerzőnél ez a szám kisebb, míg egy nyelvérzékkel kevésbé megáldott írónál lehet nagyobb.)

Ha a szóellenőrző adatbázisából kivesszük a szót, akkor ezzel nő a hibák lefedettsége, de vele nő a tévedések száma is. Ekkor 1000 előfordulás mindegyikét jelzi, ebből 990 esetben jogosan, míg a 10 helyes alak esetében téves a jelzés. Ha viszont a szóellenőrző elfogadja a szót, de környezeti ellenőrzéssel az esetleges helyes előfordulás 95 százalékban kiszűrhető, akkor jobb eredményhez jutunk, ha a szóellenőrző elfogadja az alakot, és csak a mondatellenőrző jelez hibát a környezet függvényében. Maradva az *egyenlőre* szónál, ha a környezetben találunk *hoz*, *vág*, *szab*, *tesz* igét, akkor nagyobb a valószínűsége, hogy helyes volt a szó, sőt, a statisztikák alapján a helyes előfordulások nagyobb százalékát le is fedjük. Így az 1000 előfordulásnál 990 riasztásból legfeljebb egy a téves riasztás, míg a 10 helyes esetből alig lesz hibásnak minősített. Pár hibás esetet nem veszünk észre, de legfeljebb 1-nél kapunk felesleges jelzést.

2.2. Egyszerű hibák kimutatása

Lokális ellenőrzéssel sok stiláris hibát deríthetünk fel. Például durva, obszcén kifejezéseket, vagy olyanokat, melyeket az amerikaiak inkorrekt szóhasználatnak neveznek. Magyarországon az idegen szavak szűrésére lehet igény. Ezek a szűrések nem csak arra jók, hogy stilárisan tegyük szebbé írásunkat, de időnként arra is, hogy nem kívánt elütések következtében keletkező hibákat is felderítsünk. A magyar nyelv sűrű, emiatt elütéssel könnyen keletkezhet helyes szó. Gondoljunk arra, ha a *szár* szóról lemarad az ékezet. Ettől a mondat nyelvtani szempontból még helyes marad, csak stiláris hiba keletkezik, de előfordulhat, hogy épp ettől válik nyelvileg hibássá a mondat.

Ezért is érdemes nyelvtani következetlenségeket is szűrni, bár ilyen ritkán fordul elő. Leginkább akkor, ha a szövegszerkesztővel copy-paste segítségével szabjuk-varrjuk az írást. Viszont lehetséges – és ez a gyakoribb eset –, hogy egy elütés miatt nyelvtanilag hibás (rész)szerkezet keletkezik: *A patak partján három lapút találtam.*

2.3. A lokális szintaxis alkalmazásai

A hibák egy része felderíthető lokális szintaxis segítségével[5], de időnként ez nem elég. Főként a szabad szórend miatt, de más esetekben is távol eső szavak összefüggéséből lehet rájönni a hibára.

Lokális szintaxist használnak szó-egyértelműsítésre[6][7]. Itt viszont statisztikai alapon döntenek, ami esetünkben nem jó módszer – elfedheti a hibát, ahelyett, hogy rámutatna. Magas találati aránnyal lokális szintaxist a névfelismerésnél alkalmaznak.

Hasonlóan szavak előfordulásával, illetve lokális összefüggések felismerésével operálnak szövegek minősítésekor, például a szövegek pszichológiai kiértékelésénél[8].

A nagy különbség ott van, hogy a szövegkiértékelés eredménye statisztika. Tehát a jelenségek előfordulásának mennyisége a lényeges, nem az egyedi jelenség. Emiatt ha egyszer-egyszer, akár tíz százalékban téved az eseti elemző, attól a teljes szövegstatisztika lényegesen nem módosul. A szövegkorrekciónál viszont a felismerésben elkövetett ily mértékű tévedés használhatatlanná teszi az eszközt.

2.4. A globális szintaxis alkalmatlansága

A mondatok globális formális leírása alapján hibák lokalizálását csupán a formális, például a programnyelveknél alkalmazzák, mert a jól tervezett programnyelvek egyrészt determinisztikusan elemezhetők, másrészt szintaktikus hiba esetén jól behatárolható a hiba helye. Több értelmű nyelveknél – a természetes nyelvek ide tartoznak – ez az út nem járható. Még a könnyebben elemezhető angolnál sem a teljes mondat szintaxisából indulnak ki. Helyette hibás szituációkat adnak meg, általában szűk látókörű, részszintaxis segítségével. Az egyes szabályok, szabálycsoportok függetlenek egymástól, és ha valamelyiknek eleget tesz egy részszöveg, azt hibásnak illetik, illetve az elemzésnek megfelelően javítási tanáccsal is szolgálnak.

Teljes mondatelemzőt használnak a szabályalapú fordítóprogramok. Ott viszont feltételezik, hogy a mondat helyes, illetve helytelen mondatokat is igyekszenek megelemezni, hogy lefordítsák. Emiatt az ott használt szintaxis nem pontos, alkalmatlan akár a hiba felismerésére, de a hiba pontos helyének meghatározására végképp használhatatlan.

2.5. A kompromisszum – parciális szintaxis

Miután teljes mondatelemzésről szó sem lehet, ezért lokális szintaxist, vagy ami hasznosabbnak bizonyult, a Helyesebbnél bevezetett parciális szintaxist alkalmaznak. Ebben az esetben a lokális vizsgálaton kívül lehetőség van távolabbi szavakra is rákérdezni, a mondat közbünső részét mintegy átugorva. A leírónyelv interpretálása itt is gyors, lehetőleg determinisztikusan interpretálható kell, hogy legyen. A lehetséges megoldás szavak feletti reguláris attribútumnyelvtan, mely legfeljebb sekély szintaxist használ.[1]

3. Mit ellenőrizünk?

3.1. A cél határozza meg az eszközt vagy az eszköz a célt?

Ha a cél határozza meg az eszközt, az igény az, hogy minél több – főleg gyakran elkövetett – hibát derítsünk fel. Helyesírásunknak két nehezen emészthető szabálycsoportja van, melyet sokan elvétenek. Az egyik a szóösszetétel, a másik a vesszőhasználat. A szabályok az első esetben erősen szemantikafüggőek, a másik esetben a szabályok (majdnem) egyértelműek ugyan, csak az nem világos, mitől lesz egy rész tagmondat, és mitől csak egy részkifejezés. Szóval mindkét eset nehéz dió.

Ha az eszköz határozza meg a célt, akkor az egyszerűen, de biztosan felderíthető hibákat érdemes feltárni. Az egyedi hibás esetek lehetnek ritkák, de tömegükben gyakoribbak, mint a nehezen felismerhetőek.

3.2. Mit ellenőrizhetünk?

A különböző nyelvekre megvalósított mondatellenőrzők téma szerint a következő osztályozásokat alkalmazzák:

- Nyelvtani hibák
  - Egyeztetések (alany-állítmány; névelő, szám, személy, nem; névelő hangtani egyeztetése; vonzat ...)
  - Koordinációs szavak, kötőszavak helyes használata
  - Vesszőhasználat (mondat és mondatrész vesszőhiány, -többlet)
  - Kisbetű vagy nagybetű
- Összetett szavak (amikor egybe és külön is helyesek a szavak)
  - Egybeírás (szemantikai és nyelvtani összetételek)
  - Különírás
  - Kötőjeles összetételek (akár egybe, akár külön írták)
- Gyakori elírások
  - Elütés következtében nem odaillő szó (A szóellenőrző kiterjesztése)
  - Hibás szóhasználat (A közhasználatban elterjedt hibás változat)
- Stiláris szűrés
  - Vulgáris, obszcén szavak, kifejezések
  - Inkorrekt, sértő szavak, kifejezések (Ez az, amit az USA-ban szigorúan vesznek, de magyarban is lehet találni szűrendőt)
  - Idegen szavak, kifejezések (Ha nem elterjedt szó, akkor illik átírni. Szakszövegekben nem érdemes szűrni)
  - Köznapi szöveg. (Egyes nyelveken eltér a beszélt és a szép írott nyelv.)
  - Vidékies szöveg, zsargon, rétegnyelv ...
  - Nem preferált szerkezetek - pl. angolban passzív szerkezet
- Formátumok
  - Dátum, idő és egyéb mértékek (a dátumnál a pontok használata)
  - Megszólítás
  - Címzés
- Egyebek
  - Szóközök, tabulátorok (felesleges és hiányzó szóközök szűrése)
  - Halmozott írásjelek (Ha több írásjel követi egymást, ezek nem minden szekvenciája helyes)
  - Zárójelek, idézőjelek (Egymásba skatulyázás, helyes jelek, jelpárok használata.)
  - Kötőjelek, gondolatjelek. (Nem közismert, mikor melyik jelet ildomos használni)
  - Internet formátum
  - Számformátum (pl. tizedespont, vagy -vessző)
  - Ligatúrák, alternatív kódolások (Többek között a hármaspont, lebegő ékezetek... melyek nem feltétlen látszanak az írott szövegben)

A csoportosítás és a tartalom nyelvenként és meghajtónként eltér. Egyes típusok szűrése az egyik nyelven könnyű, míg a másikon majdnem lehetetlen.

4. Technikai kérdések

4.1. A nyelvi adatbázis nyelve

A technikai feltételek lényegesek a mondatellenőrzőnél. A bővíthető eszközök nem nélkülözhetnek egy jól definiált formális nyelvet, melyen megfogalmazhatók a szabályok. A leírás nyelve nem feltétlen közvetlen vezérli az ellenőrzőt. Általában egy előzetes fordítási procedúrával keletkezett közbülső (ember által nehezen emészthető) kódban interpretálódik a végrehajtandó ellenőrzés. Ha a szótan és a nyelv szintaxisa egyszerű szerkezetű, akkor a leíró szabályokat közvetlen karakterekre épülő reguláris kifejezéssel is lehet interpretálni. A legtöbb esetben nem ezt teszik. A magyarban a leírás és a futó interpreter hivatkozik egy szóelemzőre, mely a mondatellenőrzőn belül egy külső modult képez.

A formalizmusban nem csak a hibát kell leírni. Meg kell jelölni a hiba pontos helyét és lehetőség szerint a korrekció (csere) módját is. Van, mikor a hiba felismeréséhez nagyobb területet kell átvizsgálni, mint amit érdemes a felhasználó elé tálalni. Erre is lehetőséget kell biztosítania a leírás nyelvében.

A leíró formalizmusban gyakran hivatkoznak szóalakra, szótőre, morfémákra, szótani kategóriákra. Ha elegendő a felszíni alakok felismerése, akkor általában reguláris kifejezéssel adják meg a szabályokat. Ez a ritkább eset. A nyelvi kategóriákat felhasználó szabályok is csak sekély (lapos) szintaxist használnak, bár néha jó lenne ennél összetettebb, környezetfüggetlen nyelvtan is.

Ha hiba felismeréséhez javítási ajánlat is szerepel, márpedig általában erre igény van, akkor nemcsak szótani elemzőre, hanem generálóra is szükség van. (Kivételt képeznek a gyengén vagy nem toldalékoló nyelvek.)

A szabályok kis része nem hivatkozik szóelemzésre. Ilyenkor a szabályból pontosan követhető a felismerés miként je. Az esetek többségénél praktikus a szóelemzés és az abból nyert tulajdonságokra való hivatkozás. A magyarban nem felsorolhatóak az egy kategóriának megfelelő alakok. A szóelemzőre való hivatkozás növeli a mellélövések esélyét, mivel a morfológiai modulban is lehetnek hibák. Az ilyen hibák elkerülése érdekében fontos a szóelemző pontossága. Hasonló minőségi követelmény lép fel a szógenerátorral szemben is.

A leírás nyelve olvashatónak, karban tarthatónak kell, hogy legyen, mert a szabályok száma nagy, ráadásul hathatnak is egymásra, emiatt mindig igény van javításra, bővítésre.

Tehát az egyes szabályoknak a következő információkat kell tartalmaznia:

- A hiba leírása, mely szóalakra, morfológiai egységekre, ezek szekvenciáira hivatkozik.
- A hiba hatásköre – mit kell kijelölni hibásnak a megjelenítéshez.
- Javítási javaslat: mire cseréljük a megjelenített résztartomány – ebben a felismert rész egyes elemei és az ebből generálandó szavak is szerepelhetnek.
- A hiba típusa: ez csak arra kell, hogy bizonyos osztályok ellenőrzését engedélyezhessük, letilthassuk.
- Magyarázat: mellékes információ, mely nem mellékes a felhasználónak, hogy jobban megértse, mire gondol a nyelvész.

4.2. A végrehajtó motor

Mint említettem, általában a formalizmusban leírt nyelvi információ egy fordítás után lesz használható ellenőrző. Ez vagy egy közbülső nyelv interpretálása, de nem ritka eset, amikor a nyelvtan egy része adatként interpretálódik, míg más részét a fordító közvetlenül programnak generálja. Ez utóbbira példa a csehek grammar-je[9], amit a Microsoft számára készítettek. A Lightproof is alkalmaz vegyes megvalósítást. Gyakoribb viszont az, amikor a teljes anyag interpretálható közbülső formáját használják.

4.3. Sebességi követelmény

A végrehajtásnak kellően gyorsnak kell lennie. Egy mondat ellenőrzése, legalábbis a hiba felderítése egy másodpercnél nem vehet többet igénybe. Mivel a szabályok száma ezrekben mérhető, és egy hosszabb mondatban sok helyen kell próbálkozni a lokális – parciális szabályok érvényesülésével, még a mai gyorsabb gépeken is lehet kritikus az időigény. A tapasztalat szerint jól megfogalmazott szabályok esetén a program a szótani elemzéssel tölti a legtöbb időt. Emiatt itt kell spórolni. Az elemzések gyorsítótárba való gyűjtése általában meghozza a kívánt sebességet, de nem mindig.

A morfológia megkerülése is sokat segíthet. Ha valamit tisztán alaki módon is meg lehet fogalmazni, akkor ez segít. Az is segíthet, ha a morfológiai elemzés cash-elése alatt olyan attribútumokat is letárolunk, melyek az elemzésnél gyakran kellhetnek (szófaj, eset, igerag. . .), hogy ezeket ne kelljen a morfológiai elemzésből újra meg újra kihámozni.

Ha a felismerés két részre bontható – egy durva gyorsra és egy részletesebb kiértékelésre –, akkor érdemes kettébontani. Ezzel megint csak csökkenthető a terhelés, hisz a hiba nagy valószínűséggel kizáródik az első lépésben, és ilyenkor a másodikra már nincs szükség.

Magyarban gyakori a többértelmű szó. Próbálkoztam szabályalapú egyértelműsítéssel, azt várva, hogy ha pár helyen csökkentem az alternatívákat, akkor nő a globális sebesség. Az egyértelműsítés mechanizmusa hasonló volt, mint a hibák felismerése. Többnyire lokális szabályok alapján döntöttem. A statisztika alkalmazása itt azért veszélyes, mert félreviheti a későbbi hibakeresést. A sebességnövekedés nem volt érzékelhető. Ahol szükség volt rá, inkább a hiba felismerésébe öntöttem az egyértelműsítéseket.

5. A portolás

Miután a Microsoft úgy döntött, hogy az összes nyelvi modult házon belül valósítja meg, ezen az úton a nyelvi tudásunk felhasználása lényegesen csökkent. A felhalmozott nyelvi tudást viszont nem akartam, hogy kárba vesszen. Szerencsére ekkorra már megjelent a Lightproof, melybe reményeim szerint sok mindent át lehet menteni. Mindkettő támaszkodik szóelemzőre, és leírási formalizmusukban sok a hasonlóság. Például a magyarázat rész szinte megegyezik. Ezzel nem is foglalkozom a cikkben.

5.1. A két formalizmus összevetése

A szabályok mindkét megvalósításban osztályokba sorolódnak. Az osztályok a Helyesebbnél nem módosíthatóak, a Lightproofnál adatvezérelt – viszont vigyázni kell, hogy a szabályban jelzett osztály szerepeljen a deklarációban.

A szabályok a Helyesebbnél csoportokba vannak összegyűjtve – minden csoport, mint egy fejezet először a csoport fejében deklarálja a hiba típusát, a hallgatólagos hibamagyarázatot, és típusonként tér el a hiba szkópja (kijelölendő területe) és a javításnak a módja. Például a különírási osztályban a kijelölendő mindig a hiba feje.

A Lightproof esetén minden szabály maga tartalmazza ezeket az információkat, tehát a szabály vezérli, hogy mit kell aláhúzni, és mi legyen a magyarázó szöveg.

5.2. A Helyesebb hibaosztályai

- Névelőegyeztetés – *a, e* vagy *az, ezen*
- Vesszőhiány – felesleges vagy hiányzó mondat és mondarész vessző
- Lehetséges szóösszetétel – nyelvi szabályok alapján felismert egybeírás
- Létező szóösszetétel – szótár alapján felismert egybeírás
- Kötőjeles szóösszetétel – ha kötőjellel kell egybe írni
- Hibás szóösszetétel - egybe írt, de külön szóba írandó
- Helyi szerkezeti hiba, hiány – legtöbbször valamilyen alany esetű rész hiánya
- Téves szóhasználat – amit sokan rosszul tudnak
- Trágár szavak szűrése
- Durva szavak, kifejezések szűrése
- Idegen szavak szűrése
- Általános szerkezeti hiba – ilyet is felismerek, de ritkán
- Nagy kezdőbetűvel írandó – mondat eleje, tulajdonnevek
- Kis kezdőbetűvel írandó
- Szóközök, írásjelek ellenőrzése – halmozott szóköz, halmozott írásjelek, tapadó írásjelek

Ezeket az osztályokat minden gond nélkül deklarálhattam a Lightproofban. Van még két rejtett osztály. Az egyik a hagyományos helyesírás-ellenőrző – ezt a Microsoft nem használta – és egy jelzés túl hosszú mondat esetére.

5.3. Lényegi különbségek a formalizmusban

A szabályok szintaxisa hasonló. Egy szabály négy részből áll: gyors felismerési minta, egyéb feltételek, javítási opciók, magyarázat. Azonban funkcióban a részek eltérnek.

rész	Helyesebb	Lightproof
fej, primer ellenőrzés	egyetlen szó: szóalak, morféma(sorozat), kategóriafeltétel	akár több szavas karakteres reguláris kifejezés
további, másodlagos feltételek	szóalakot, HUMOR kategóriákat használó (szóalak, nyelvi tulajdonságok, morfémaszekvenciák) több irányú reguláris kifejezés	az osztályra, a fej reguláris részkifejezéseire és Hunmorph elemzésekre vonatkozó kifejezések a környezeten belül, ezek logikai kifejezései
feltétel elemeinek szekvenciája	a szintaxisban tetszőleges irány és szekvencia kijelölhető	csak jobbról balra szabályok, illetve közvetlen környezeti elemek
távolhatás	irányított, tetszőleges	irányítatlan, reguláris
hiba kijelölése	osztálytól függő, program vezérli a másodlagos feltételek szekvenciája alapján	kijelölhető a fej reguláris kifejezése alapján
javítási ajánlatok	kulcsszavak, hivatkozás a megtalált részekre, generálás a fej elemzése alapján	kulcsszavak, hivatkozás a megtalált részekre, generálás minta alapján
generátor	képzőket is	csak ragokat, jeleket
magyarázat	közvetlen magyarázat és opcionálisan részletes magyarázatra való hivatkozás, mely nem itt, hanem ettől függetlenül tárolódik – lehet hivatkozás a MHSZ-ra	közvetlen magyarázat és opcionálisan részletes magyarázat – hivatkozás weboldalra, legtöbbször a MHSZ-ra
formális hiba esetén	teljes konzisztencia ellenőrzés fordításkor	nem minden hiba szűrhető ki fordításkor

5.4. Különbségek a motorban

Helyesebb	Lightproof
8 bites kód	UNICODE
C-ben megírt interpreter	Python-alapú interpreter
Felhasználó nem bővítheti rutinokkal	Felhasználó bővítheti P rutinokkal
Prioritás az osztályok között	Nem ismert a prioritás vezérlése
Szabálykölcsönhatás kiküszöbölése	Nincs rá lehetőség

5.5. A portolás menete

Kiindulva a működő Lightproof szabályokból, először módosítottam a hibaosztályokat és a hozzá tartozó dialógust, a Helyesebb szabályokat megkíséreltem konvertálni Lightproof szabályokra. Először az egyszerűbbeket, majd az egyre összetettebbeket.

Példa kézi kódolásra, mikor a reguláris kifejezéssel biztosan megoldható a találat:

Helyesebb – a négy sort kézi kódolással eggyé lehet alakítani

```
hülyéskedik[IGE]
hülye[MN]
hülyít[IGE]
hülyül[IGE]
```

Lightproof – reguláris kifejezés, nem olvasmányos, nem használ morfológiát

```
[-\w]*hüly[eéüí][-\w]*<-option("offensive")->_
```

A további szabályokat PERL-ben megírt programmal konvertáltam. Néhány példa:

Helyesebb – itt nem kell morfológia (az összetételt a motor generálja)

```
már < mint pedig
```

Lightproof – itt mindig meg kell adni a javaslatot is

```
már (mint|pedig) <- option("compound") -> már\1
```

Helyesebb – egy egyszerű sorból

```
csesz[IGE] &-Csesznek
```

Lightproof – nehezen, de olvasható kifejezés keletkezik

```
[\w]*cses+z[\w]*<-option("offensive")and morph(\0,r"st:csesz(ik)?\b",False)->_
```

Helyesebb – a képzett alakokra is működik

```
zsáner[FN](osszetett=0) = kedvenc[FN] ízlés[FN] modor[FN]
```

Lightproof – sajnos a visszatoldalékolás nem megy képzőkre

```
zsáner\w*<-option("foreignwords")and morph(\0,r"st:zsáner\b",False)
and not morph(\0,r"st:zsáner po:noun ts:NOM pa:",False) ->=
generate(u"kedvenc",\0)[0] + "|" + generate(u"ízlés",\0)[0] + "|" +
generate(u"modor",\0)[0]
```

Helyesebb – ez már kicsit összetettebb

```
csatlakozó[MN](osszetett)> { alig eléggé apránként gyorsan lassan hamar
nem } > (nevszoi)(IGEMIN+rag=ALL) = csatlakozó_
```

Lightproof – a hatás pontosan megegyezik

```
((alig|eléggé|apránként|gyorsan|lassan|hamar|nem) )*((csatlakozó)(\w+))
<- option("devide") and morph(\3, r"po:vrb ts:PRES_INDIC_INDEF_SG_ 3
ds:Ő_PRESPART_adj ts:NOM pa:",False) and word(-1) and (morph(word(-1),
r"is:ALL", False))) -3> \4 \5
```

Helyesebb – itt tipikus távolhatás van

```
mintha <{(Felt=0) (igei)(IGEMIN=0)}<(igei)(IGEMIN+Felt=IGEMIN) = mint_ha
```

Lightproof – a távolhatást itt is majdnem tökéletesen meg lehetett oldani

```
mintha (?![\w]+n([ae]|á(|[mdk]|na?k|tok)|é(|[mdk]|nek|tek)),? )*
<- option("devide")and not(word(1)and morph(word(1),
r"PRES_COND_[SP][GL]_[123]", False)) -> mint ha
```

Helyesebb – itt egyszerű távolhatás van

```
produkál[IGE] >< zajt ricsajt lármát = csap[IGE]
produkál[IGE] ->< zajt ricsajt lármát = létrehoz[IGE] felmutat[IGE]
megteremt[IGE] alkot[IGE] termel[IGE] gyárt[IGE]
```

Lightproof – a távolhatás tökéletesen azonos

```
produkál(\w*) <- option("foreignwords") and stem(\0) == ["produkál"]
and (zajt|ricsajt|lármát) in TEXT ->= generate(u"csap", \0)[0]
[-\w]*(produkál\w*) <- option("foreignwords")and morph(\0,r"st:produkál
po:vrb",False)and not morph(\1,r"pa:",False)and not(zajt|ricsajt|lármát)
in TEXT -1>= generate(u"létrehoz", \1)[0] + "|" +
generate(u"felmutat",\1)[0] +"|"+ generate(u"megteremt" \1)[0] +"|"+
generate(u"alkot",\1)[0] +"|"+ generate(u"termel",\1)[0] +"|"+
generate(u"gyárt",\1)[0]
(produkál)\w+<-option("foreignwords")and morph(\0,r"st:produkál po:vrb",False)
and morph(\0,r"pa:",False)and not(zajt|ricsajt|lármát)in TEXT -1> létrehoz
|felmutat|megteremt|alkot|termel|gyárt
```

Mint a példákból is látható, van olyan szabály, melyet több szabállyá fordítottam, de általában egy sorból egy sor keletkezett. Ahol lehetett, kerültem az elemzést és a generálást. Mintegy háromezer-ötszáz sort sikerült lefordítanom. Ezek többsége lokális szintaxissal megfogalmazható volt. Ilyen a stiláris hibák többsége. Szám szerint ezek a szabályok adják a zömét, de nem mindet.

5.6. Amit nem tudtam átvinni

A két kritikus szabálycsoport a magyarban: az egybe- és különírás, valamint a vesszőgondok. Mindkettőnél komoly akadályba ütköztem.

A szóösszetétel szabályainak többségét sikerült átvinni, de a több esetet lefedő általános szabályoknál nem működött kielégítően. Ennek fő oka a szóelemző adatbázisa volt.

Az összetételek általános szabályaiból: két főnév – ha szerepel ige is a mondatban –, általában egybe írjuk, ha az első ragozatlan, jelöletlen, a másodiknak meg nincs birtokragja.

A Helyesebb algoritmusában ezt akkor alkalmaztam, ha az összetétel szerepel szótári tételként a morfológia szerint. Bár a Hunmorf is tartalmaz ilyen tételeket, de nem olyan jó válogatást, mint a MUMOR szótár. További korlát, hogy az általam használt HUMOR szótárban finomabb szófaji beosztások vannak. A főnév és melléknév közti szófajok (népnév, foglalkozásnév...) melyek módosítják az összetételi szabályt, szerepelnek a szótárban. Ha a szótan nem pontos, akkor bizonytalan a szóösszetételek ellenőrzése.

Máshol az egyéb feltételek mellett csupán az ellenőriztem, elfogadja-e a Hunspell az összetételt. A Hunspell a HUMOR-hoz képest nagyon megengedő.

A másik nehézség a vesszőknél támadt, Itt sok szabályban szerepelt olyan feltétel, melyet nem vagy csak korlátosan lehetett megvalósítani a Lightproofban. Tipikus eset, a ragozott ige meglétének keresése.

Például, ha két ragozott ige szerepel távol egymástól, és nincs köztük írásjel, akkor biztos kifelejtették a vesszőt. Erre számos szabály van a Helyesebb-ben. Azért számos, hogy a hiba valószínű helyét is felderítsem. Ezek egyike sem vihető át korlátozás nélkül, mert távolhatást legfeljebb reguláris kifejezéssel adhatok meg, de míg a Helyesebbnél ebben absztrak tulajdonságok is szerepelhetnek, a Lightproof csak formai (karakteres) reguláris kifejezéseket enged meg.

5.7. A Hunspell hibái

Gondot okozott a morfológia pontatlansága is. A HUMOR adatai közt is van hiba – hibátlan nyelvi adatbázis nem létezik –, de a tapasztalatom szerint a Hunspell adatai közt nagyságrenddel több tévedés található. Bár a szótár sok fésülésen ment keresztül, de visszajelzések eddig csak a helyesírás-ellenőrző használatából keletkeztek. A melléelemzések ezt nem zavarták, de a további elemzésnél már zavaró volt. Hát még a hibás generátumok.

Gyors viselkedésbeli vizsgálatom alapján arra a következtetésre jutottam, hogy a szótár építésénél túl sok valószínűségi alapon születő döntést hoztak emberi beavatkozás nélkül. A forrásanyaggal sajnos eddig nem volt szerencsém találkozni, de világossá vált, hogy gyakran dönthettek formai alapon. Így az *az* szónak – lett légyen névelő vagy mutató névmás – allomorfja az *azt, annak, abból...*, de sohasem az *arra, arról...*. Bár az allomorfokkal nem foglalkoztam, de nem szabad ilyen gyakori esetekben tévedni.

Hasonlóan érdekes, hogy a *mag* szónak a többes száma lehet *magak*. Feltehetően a *mag* főnév alakjait keresve szép számmal találtak olyan alakokat, mint *magas, magam, magad...*, emiatt ha alaki statisztikát alkalmaztak, bejött az ajakkerekítés nélküli toldalékolás is.

Az összetételeknél nem elég kifinomultak a szabályok. Itt is nagy előnyt jelent a HUMOR-nál, hogy sok célra használták már, és ezektől sok minden finomodott a morfoszintaxisban.

5.8. A kölcsönhatások kezelése

Kevés szabálynál ez nem volt lényeges, de nagy számú szabályok esetén elengedhetetlen, hogy kezeljük azok kölcsönhatását. A Helyesebb-ben ez a szóösszetételnél vált nélkülözhetetlenné. Ezzel kapcsolatos, hogy irányíthatóvá kell tenni a szabályok prioritását is. (Szabályok kölcsönhatását erősen szabályozza a MetaMorpho fordító. Ott a szabályok száma több nagyságrenddel felülmúlja a mondatellenőrzőét[10].)

6. Tervek a jövőre

A megszületett ellenőrző minden hibája ellenére sokkal jobb, mint a korábbi. Az eredeti Lightproofban pár száz gondosan megfogalmazott szabály szerepel. Ezek a hibák kis százalékát derítették fel némi mellélövéssel. Most közel 5000 szabály van, mely nem fedi fel a hibák felét, de kisebb a mellélövések aránya. Biztató kezdet, de nem elég.

A szűk keresztmetszet a Hunspell adatbázisa. Nem a mérete, hanem módszeres javításra szorul. Addig nem érdemes tovább lépni.

A másik szűk keresztmetszet, hogy lehetőség legyen a szavak tulajdonságait is bevenni a reguláris kifejezésekbe. Ehhez Python-bővítésre van szükség, amit a Lightproof megenged, de ezt eddig nem alkalmaztam.

Ha fejlődő rendszerben gondolkodunk, akkor mindenképpen szükség van egy olyan formalizmusra, ami olvasmányos. Ebből egyszerű konvertálással kell megkapni a Lightproofnak megfelelő formalizmust. A jelenlegi szintaxis nem alkalmas az adatbázis fejlesztésére, mert a szabályok nehezen olvashatóak, és formai hiba esetét nem deríti fel a fejlesztő rendszer, futás időben akár fenn is akadhat. A változtatással viszont várható, hogy olyan dinamikusan fejlődő eszközt kapunk, mint a franciák eszköze[11], amely a Lightproofból indult ki, de sokat bővült, és nap mint nap fejlődik a közösség segítségével.

Hivatkozások

1. Naszódi, M.: Nyelvhelyesség-ellenőrzés számítógéppel: Parciális szintaxis[1]. In: VII. Országos Alkalmazott Nyelvészeti Konferencia., Budapest (1997)
2. Németh, L.: A Lightproof lehet a nemzetközi OpenOffice.org nyelvi ellenőrzője[2] (2009)
3. Naszódi, M.: Statisztika megbízhatósága a nyelvészetben[3]. In: XI. Magyar Számítógépes Nyelvészeti Konferencia, Szegedi Tudományegyetem (2015) 34–45
4. Naszódi, M.: A magyar helyesírás-ellenőrzők mai állása[4]. In: XIII. Magyar Számítógépes Nyelvészeti Konferencia, Szegedi Tudományegyetem (2017) 347–354
5. Verberne, S.: Context-sensitive spell checking based on word trigram probabilities[5]. Master's thesis, Taal, Spraak & Informatica University of Nijmegen (2002)
6. Hanák, P.: Magyar nyelvi szófaji egyértelműsítő módszer fejlesztése gépi tanulási algoritmusok felhasználásával[6]. (2002)
7. NyI: Magyar Nemzeti Szövegtár[7] (2002)
8. Ehmann, B., Garami, V., Naszódi, M., Kis, B., László, J.: Subjective Time Experience: Identifying Psychological Correlates by Narrative Psychological Content Analysis.[8]. (2007) 14–25
9. Kuboň, V., Holan, T., Plátek, M.: A Grammar-Checker for Czech[9]. In: ANLC '97 Proceedings of the fifth conference on Applied natural language processing. (1997) 147–154
10. Prószéky, G., Tihanyi, L., Ugray, G.: Moose: a robust high-performance parser and generator.[10]. In: Proceedings of the 9th Workshop of the European Association for Machine Translation, La Valletta, Malta (2004) 138–142
11. WWW: GRAMMALECTE correcteur grammatical open source[11] (2012)

[1] http://www.mek.iif.hu/porta/szint/tarsad/nyelvtud/nylvhely/nylvhely.htm
[2] http://libreoffice.hu/a-lightproof-lehet-az-openoffice-org-alapertelmezett-nyelvi-ellenorzoje/
[3] http://rgai.inf.u-szeged.hu/mszny2015/files/MSZNY2015_press_B5_PQ.pdf
[4] http://www.cs.bme.hu/~naso/langeng/SpellsSate20016.pdf
[5] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.324.7146&rep=rep1&type=pdf
[6] http://members.iif.hu/hanak/prj/i3/i027/ds/hu/i027-dshu.pdf
[7] http://mnsz.nytud.hu/
[8] http://www.cs.bme.hu/~naso/langeng/Ehmann_etal_Empirical_2007_3.pdf
[9] http://www.aclweb.org/anthology/A97-1022
[10] http://www.morphologic.hu/downloads/publications/tl/eamt_2004_malta_pg-tl-ug.pdf
[11] https://www.dicollecte.org/

Conclusions from the Conversion of Linguistic Data of a Hungarian Grammar Checker

Mátyás Naszódi, e-mail: naszodim@morphologic.hu

MorphoLogic, 1122 Ráth György utca 36. Hungary

Extended abstract

Grammar checkers are unable to decide whether a sentence is grammatically correct or not. They only look for faulty parts in the sentence that are hidden from spell checking. In most cases, proofreaders use local syntax (like in context sensitive spelling) based on probability of sequences of words.

The quality of proofing tools depends on the applied formalism for describing the linguistic database. Its goodness is measured by the percentage of recognized real errors and by the rate of false alarms. The applied formalisms and drivers have limited capabilities.

Microsoft decided to develop its own language support for Office. The last externally developed tool, the grammar checker *Helyesebb* (by MorphoLogic) was replaced by their own engine and data a year ago. The quality of the new proofing tool is below the acceptable level. I have tried to port the linguistic knowledge of Helyesebb to LibreOffice, as an open source code. By that time, the grammar engine *Lightproof* (by László Németh) had been developed however, it contained only a small amount of linguistic knowledge.

This article focuses on the capabilities of Helyesebb and Lightproof, and it demonstrates how it is (im)possible to convert linguistic knowledge from one formalism to the other one. The experiment also provides other linguistic conclusions. It points to the quality problems of morphological descriptions and shows some linguistic phenomena concerning proofing modules.

The formalisms of the linguistic database of the two tools are similar. More than 4000 rules which use only neighborhood checking have been transformed with the help of a PERL program. Handling far relations however needs *partial syntax* instead *local*.

Another conclusion is that the morphological system (in open source systems it is *Hunspell*) is enough for spelling, but it may not satisfy the needs of analysis and word generation requirements for grammar checking.

The obtained quality is much better than that of the original Lightproof, but is far from the desired. The bottlenecks of the development are the weakness of morphology, the lack of possibility of usage of far relations, and the low level formalism of the linguistic description.