PureToken: egy új tokenizáló eszköz

Indig Balázs[1]

[1]Pázmány Péter Katolikus Egyetem, Információs Technológiai Kar,
MTA-PPKE Magyar Nyelvtechnológiai Kutatócsoport
1083 Budapest, Práter u. 50/a
indba@digitus.itk.ppke.hu

Kivonat A szövegek mondatra és tokenekre bontása manapság már nem aktív terület, így a rendelkezésre álló eszközök, amelyek ezt a feladatot végzik, a karbantartás és fejlesztés hiányától szenvednek. A jelenleg rendelkezésre álló, mondatra és tokenre bontó legjobb magyar eszköz, a Huntoken fejlesztése régóta nem aktív, viszont számtalan projektben van szükség egy ilyen eszközre. A Huntoken készítésekor a kornak megfelelő technológiákat alkalmaztak a szerzők, mint például a Latin-2 karakterkódolás és a Flex lexikaielemző-generátor. Ezek a technológiák mára elavultak, és átvette a helyüket más, például a Unicode-karakterkódolás. A jelen tanulmányban bemutatunk egy eszközt, amely a Huntoken alapjaiból kiindulva és a részletes specifikációs teszteket felhasználva, azzal teljesen azonos kimenetet képes generálni, ám Unicode alapon. Bemutatunk egy olyan változatot is, amely egy beépített morfológiai elemzőt (a Humort) felhasználva kisebb méretűvé válik, viszont egy átláthatóbb megoldáson alapul. Ez a rendszer – bár nyelvfüggő, de – a más nyelvekre való jó minőségű kiterjeszthetőség lehetőségét is magában hordozza. Rövid távú célunk, hogy a létrehozott új eszköz sebességében és a kimenet minőségében is megegyezzen, sőt hogy meghaladja a Huntokent. Az első mérések egy kimondottan erre a célra összeállított korpusszal készülnek.

1. Bevezetés

2003-ban az első Magyar Számítógépes Nyelvészeti Konferencián bemutatták a Huntoken szabályalapú mondatra bontót és tokenizálót [1]. Az akkori mérésekből kiderült, hogy a szabályalapú módszer sokkal hatékonyabb, mint a statisztikai gépi tanuláson alapuló változatok. Akkor a program 99,03%-os hatékonyságot ért el a Szeged-korpuszon [4]. Ezzel a problémakört megoldottnak tekintette mindenki, és a program fejlesztése abbamaradt. Az azóta eltelt majdnem 10 évben az informatika és a természetesnyelv-feldolgozás is változott, a Huntoken mára majdnem minden nyelvtechnológiai alkalmazás előfeldolgozó programjává vált, pusztán a hatékonysága miatt, ezáltal egy nélkülözhetetlen eszközzé nőtte ki magát.

Eközben az informatikai fejlődés sok szempontból elavulttá tette és az így jelentkező problémák áthidalása egyre több munkát okozott mindenkinek. Eljött

az idő, hogy aktualizálják a programot, de úgy, hogy az a jelenlegi hatékonyságát is tartsa meg. A program számtalan apró technikai változtatáson esett át és néhány újabb időközben megjelent mintát is kapott. Ezek rövid bemutatásra kerülnek a következő fejezetben.

2. Változások a Huntokenhez képest

A Huntoken alapját a GNU Flex lexikaielemző-generátor adta, amely a programozási nyelvek területén egy nagy múltra visszatekintő eszköz. Az elsődleges felhasználási területe a programozási nyelvek, melyek az angol nyelvből merítenek ihletet, így a világban bekövetkező nemzetközi trendekre érzéketlen. Ez az oka, hogy nem támogatja az Unicode karakterkódolást, ami majd 10 év alatt de facto internetes szabvánnyá vált a Latin-karaktertáblák helyett. Erre irányuló fejlesztési törekvések nincsenek is tervben a kicsiny igények miatt. Ez már a Huntoken használatánál is plusz munkát eredményezett[1]. A Flex alapot le kellett cserélni egy másik, hasonló programra. A választás a Quex nevű lexikaielemző-generátorra [3] esett, aminek elsődleges célkitűzése az Unicode támogatása a lexikális elemzésben. A program aktív fejlesztésnek örvend, sokan használják a tudása és a gyorsasága miatt, így kiváló új alapot teremt a Huntoken átiratának. A Quex Python-alapú elemzővel C vagy C++ forráskódot képes generálni, a Flexéhez nagyon hasonló felépítésű fájlokból. És ezzel teljesen platformfüggetlen ellentétben a GNU Flex-szel.

Az egyes szűrők szükség szerint át lettek csoportosítva a következőképpen:

- a *latin1* és *clean* szűrők összevonásra kerültek a *clean* szűrőbe
- a *abbrev* és a *abbrev_en* szűrők összevonásra kerültek
- a *sentbreak* szűrő törlésre került
- az *abbrev* szűrőből kikerültek a nem oda való korrekciós minták
- a token szűrő szétszedésre került több logikai részre.

Az egyes szűrők működésében is felléptek változások. A *clean* szűrő a lehető legtöbb entitást felismeri és visszaalakítja a Unicode megfelelőjére. Bemutatásra került egy új szűrő, az *escape*, amely azért felelős, hogy a mezőelválasztó karaktereket levédje olyan módon, hogy lehetőség szerint HTML-entitássá alakítsa[2].

Az *abbrev* szűrő felhasználta az M4 makrógenerátort, illetve egy Bash scriptet a rövidítésfájlok feldolgozására és a rövidítések a Flex fájlba, mintaként történő beillesztésére. Ennek a mechanizmusnak több hibája is volt, amelyek javításra kerültek. Ezeket röviden ismertetem:

- A rövidítésfájlok duplikációkat tartalmaztak. Ezek kétszer kerültek be a belőlük generált mintába.
- A rövidítéseket tartalmazó minta végére lezáróelemként a „nyug.” rövidítést mindenképpen beillesztette.

[1] Lásd clean és latin1 szűrők.
[2] Alapesetben ez a kacsacsőr jeleket érinti.

– Bizonyos mennyiségű (kb. 100-nál több) rövidítés esetén a program nem volt hajlandó lefordulni.
– A forrásfájl tartalmazott beégetett rövidítéseket, mint például a „CD”, amelyek kivételként kezelendők. (Mert gyakoribb esetben mondatvégek, és nem rövidítések.) Ez a lista nem volt bővíthető.
– Több különálló rövidítésfájl nem volt alkalmazható egyszerre.

Ezt a feladatot az új verzióban egy Python script végzi el, a fentiek figyelembevételével. Az *abbrev_en* szűrő egyedüli angol nyelvű szűrőként állt és csak egy tesztben és a rövidítéslistában különbözött az *abbrev* szűrőtől, ezért megszüntettem. Az angol nyelvű szövegek tokenizálásáról a következőkben lesz szó.

Egységesítésre kerültek a szűrőkben felvett definíciók, különös tekintettel a karakterosztályok neveire. Ennek célja, hogy nyelvfüggetlenebb legyen és többnyelvű[3] környezetben is képes legyen működni néhány definíció átírásával.

Az egységesítés lehetővé tette továbbá, hogy az eredeti XML-formátumtól eltérő, szabadon választott mezőelválasztó-jeleket lehessen használni, így mostantól ez a lehetőség is adott. A Unicode karaktertábla nagysága miatti szükséges változtatásként bevezettem, hogy csak egy meghatározott síkkészletet használjon a program, így a generált elemző kisebb és gyorsabb lesz. Ez a joker karakter („.” reguláris kifejezés), illetve a karakterlista-negáció („[ˆABC]” kifejezés) esetén fontos, mert ezeknél nagyon megnő a generált automata állapotszáma. Ennek elkerülésére a kiválasztott Unicode-síkok uniójával el vannak metszve ezek a kifejezések, és használjuk őket a későbbiekben. Ez a Quex beépített funkcióival valósult meg.

Az informatikai kifejezések között újak jelentek meg, mint például az IPv6 szabvány, vagy az ékezetes és Unicode-karaktereket tartalmazó, tetszőleges TLD-re végződő doménnevek. Ezek mind jobban bekerülnek a köztudatba, így az internetcímekkel kapcsolatos mintákat kibővítettem ennek megfelelően. Így már ezeket a tokenosztályokat is felismeri. Az egységesítés során a minták átnézése, javítása, egyszerűsítése is megtörtént. Ez főleg a tokenszűrőt érinti. A változások követéséhez, a Huntokenhez mellékelt Holt lelkek című Gogol-művet is felhasználtam, amit beépítettem állandó tesztnek a rövid specifikációtesztek mellé.

A tokenizálás nyelvfüggetlenségének érdekében két független verzió is készült az eredeti, csak minimális, a Quexre történő átültetéshez engedhetetlenül szükséges változtatásokat tartalmazó változat mellett, aminek célja az eredeti Huntoken funkcionalitások minél hűbb megtartása a Unicode karakterkódoláson.

A további két verzió egyike tartalmazza a fent említett változásokat, illetve a nyílt tokenosztályok ragozásának elemzésénél térnek el: az egyik változat megtartja az eredeti Huntokenben használt ragozásfelismerő eljárásokat és az MSD-kódolást. A másik változat egy beépülő morfológiai modulnak helyet ad, amely a beadott szó alapján meghatározza a lemmát és a címkéket. Ha más nyelven akarnánk használni a mondatra bontót, a megfelelő morfológia beépítésével erre is lenne lehetőségünk, mivel a legtöbb nyelvben már elérhető jó minőségű

[3] Itt elsősorban az európai nyelvekre gondolok. Például angol, német stb.: ezek speciális szóalkotó karaktereket tartalmazhatnak.

morfológiai elemző, de ezt az esetet gyakorlatban nem vizsgáltuk. Opcionálisan ez a lépés kihagyható, így a tokenekre bontás elemzés nélkül hajtódik végre, lehetőséget adva az utólag külön menetben történő elemzésre. Maguknak a tokeneknek a morfológiai elemzése elvégezhető lenne a tokenizálással egy menetben, de jogi okok miatt a Humor elemzőt [2] jelenleg még nincs mód beletenni a nyílt forráskódú rendszerbe.

A nyelvfüggetlenség mellett a különböző szakterületekre való könnyebb adaptálhatóság is a célok között volt. Például az orvosi szövegekben rengeteg rövidítés található, ellenben kevés az internetes cím.

A Quex képességeinek köszönhetően megoldható a különböző szűrők egy programba való lefordítása, illetve a szűrőnként az elemző bemenetének pufferből történő adagolása, ezzel további új távlatokat nyitva a program szélesebb körű felhasználhatósága előtt.

3. Eredmények

A cikkben bemutatott program, a PureToken egy olyan platform, nyelv- és címkekonvenció-független, Unicode-alapú mondatra bontó és tokenizáló eszköz, amely az elődjéhez képest számos kibővített funkciót tartalmaz, a kor elvárásainak megfelelően. Gyorsaságában és pontosságban ugyanazt a teljesítményt nyújtja, mint elődje, de néhány új tokenosztályt is felismer, és sokkal jobban testre szabható a működése. Az első tesztek is ezt igazolják. Egyetlen függősége a Quex- és a Python-környezet, valamint a C++ fordító. Célom, hogy széleskörű tesztelés után a visszajelzések alapján az esetleges új, vagy már a Huntokenben is meglevő hibákat javítsam, és szükség szerint karbantartást végezzek a programon, hogy minél több alkalmazási területen megállja a helyét.

Köszönetnyilvánítás

Köszönöm Németh Lászlónak, hogy megírta és szabadon hozzáférhetővé tette a Huntokent és hogy a fejlesztés során rendkívül hasznosnak bizonyult specifikációteszteket is írt hozzá.

Köszönjük a TÁMOP-4.2.1.B – 11/2/KMR-2011–0002 projekt részleges támogatását.

Hivatkozások

1. Miháczi A., Németh L., Rácz M.: Magyar szövegek természetes nyelvi feldolgozása. In: Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2003). Szeged (2003) 38–43
2. Prószéky, G., Novák, A.: Computational Morphologies for Small Uralic Languages. In: Inquiries into Words, Constraints and Contexts. Stanford, California (2005) 150–157
3. http://quex.sourceforge.net/ Elérés 2012. 11. 30.

4. Csendes D., Hatvani Cs., Alexin Z., Csirik J., Gyimóthy T., Prószéky G., Váradi T.: Kézzel annotált magyar nyelvi korpusz: a Szeged Korpusz. Magyar szövegek természetes nyelvi feldolgozása. In: Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2003). Szeged (2003) 238–247