Автоматична граматична корекция на българския език

Проектът се разработва между колектив от Секцията по компютърна лингвистика от
Института за български език при БАН и екип от Института за чешки език при ЧАН.

срок – 01. 01. 2008 - 31. 12. 2010

1. ТЕОРЕТИЧНИ ОСНОВИ НА РАБОТАТА ПО ПРОЕКТА[1]

Във формалния синтаксис естественият език се представя като множество от поредици от думи (словоформи). Кои поредици принадлежат на езика се определя от някаква (формална) граматика, т.е. от математически механизъм (напр. контекстно-свободната граматика), който свързва една езикова поредица с нейната структура. По този начин, наличието на структура различава поредиците, принадлежащи на езика от "не-изреченията" (т.е. поредиците, на които не може да бъде приписана (коректна) синтактична структура). Поради това задачата за идентифицирането на принадлежността на една поредица към някакъв език и задачата за определянето на структура за дадена поредица се разглеждат като идентични и като такива не се разделят (а всъщност и не могат да бъдат разделени, поради тяхната идентичност). На свой ред, обаче, подобен подход има недостатъка, че той (поне на формално ниво) не допуска не(гативни) структурни ограничения. Един тривиален (опростен) пример на такива не-структурни ограничения е възможността да се постави на преден план изискването, че в нито едно изречение в български предлог не може да бъде директно следван от лични глаголи[2]. Лесно е да се види, че когато една поредица нарушава дадено ограничение, то тя не може да бъде правилно изречение в български, но и че това ограничение очевидно не дефинира никакво структурно описание за поредицата[3]. В тясна връзка с това е също и стандартната пресупозиция, че границата между поредиците, които са граматични, и тези, които не са граматични, е ясна и рязко очертана (за схема вж. Фиг. 1).

Дори елементарната езикова практика (напр. като носител на езика или преподаването на роден език) убедително показва, че тази пресупозиция не е валидна за естественото понятие „граматичност", разбирано като приемливост от носителите на езика.

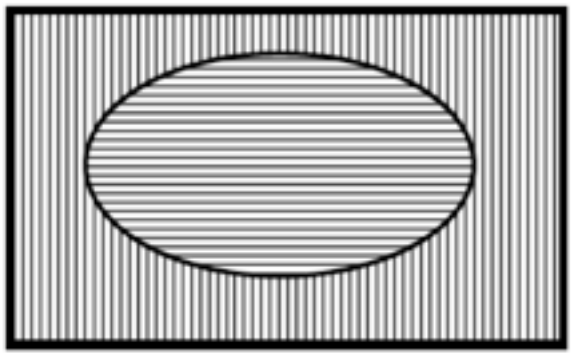
---

1 Текстът е публикуван.

2 Цитати и подобна употреба (като напр. *Думата 'в' е предлог.*) очевидно трябва да бъдат изключени.

3 Това е умишлено опростен пример, тъй като ограничението за невъзможността предлог да бъде непосредствено следван от личен глагол в български е изводима – дори когато не е в опростен и недвусмислен вид – от "стандартните" ограничения. Следващите примери, обаче, ще покажат, че съществуват граматични ограничения, неструктурни по своята същност, които не могат да бъдат изведени от "структурни" ограничения. При това ще бъде демонстрирано, че неструктурните ограничения ─ изводими от структурни ограничения или не ─ имат по-голяма практическа стойност при приложенията.

Всички възможни поредици



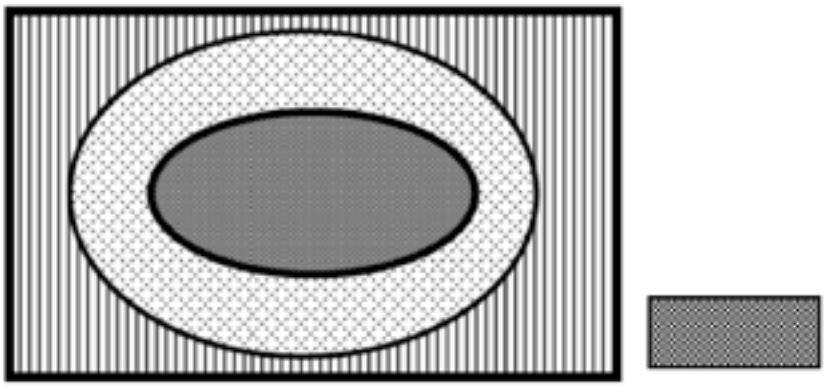
правилни поредици (изречения)

неправилни поредици

Фигура 1. Възможни поредици

От тази перспектива реалистичната картина прилича по-скоро на схемата във Фиг.2: има поредици, които се смятат за несъмнено правилни („граматични”) от носителите на езика, има други, които са недвусмислено неправилни (извън езика, "неофициално неграматични", неприемливи за носителите на езика), и има едно съвсем не незначително множество от поредици, чийто статус по отношение на коректността (приемливост, граматичност) не е изцяло ясен и/или мненията на носителите на езика се различават (някои вероятно клонящи повече в едната, а други, по-скоро в другата посока, и др.).

Всички възможни поредици



"явно" правилни поредици

"явно" неправилни поредици

поредици с несигурен /неясен граматичен статус

Фигура 2. Реални поредици

Имайки предвид по-добрата адекватност на картината във Фигура 2, целта на тази статия е да се допълни стандартното описание на синтаксиса на естествения език по начин, който да отразява лингвистичната реалност по-вярно от стандартния модел. Това ще бъде направено чрез дискутиране на методи за откриване и описание на "негативни правила" от "негативната граматика". В това отношение следните две положения могат да бъдат постулирани като начало за теоретичните проучвания:

1. граматичност / неграматичност се дефинира за цели изречения (т.е. не само за части от изреченията, поне не в общия случай)
2. неграматичността се появява (само) като резултат от нарушаването на някакво лингвистично явление/явления в рамките на изречението[4].

Тъй като всяка една „явна" грешка съдържа нарушаване на някакво езиково явление, то изглежда основателно търсенето на неправилните конфигурации да бъде предшествано от една поне приблизителна квалификация на явленията.

От гледна точка на начина на тяхната проява синтактичните явления (зависимости) могат да бъдат разделени на три класа: селективни, словоредни и съгласувателни.

Селективни зависимости: в едно доста широко разбиране селектирането (като обобщено понятие на субкатегоризацията) е изискването даден елемент (синтактична категория, понякога дори отделна дума) E1 да се появява в изречение само ако присъства друг елемент E2 (или множество от елементи {E2, E3,…,En}), т.е. ако E2 (или {E2, E3,…,En}) присъства(т) в една поредица, но E1 не присъства, то съответният случай на селективно явление е нарушен и поредицата трябва да се смята за неграматична.

Пример: ако глагол от типа рефлексива тантум се появи заедно със своя субект, то тогава рефлексивната частица (*се* или *си*) също трябва да присъства в изречението (виж напр. контраста в граматичността между поредиците *Мария щеше да му* се *е била обадила.* и **Мария щеше да му е била обадила.*).

Словоредни явления: Правилата за подреждане на думите (словоредът) са правила, които дефинират реда при подредбата на (два или повече) елемента E1, E2, …, присъстващи в рамките на определена поредица; ако този ред не е спазен, то съответното словоредно явление е нарушено и поредицата трябва да се смята за неграматична.

---

4 Строго погледнато една неграматична поредица не би трябвало да бъде наричана изречение.

Пример: редът на клитиките в групата на клитиките е строго фиксиран и всяка една промяна в този ред води до неграматичност, сравни напр. **Мария щеше да се му е била обадила.* с *Мария щеше да му се е била обадила*.

Съгласувателни явления: отново казано доста общо, явлението съгласуване изисква, ако два (или повече) елемента Е1, Е2, … присъстват едновременно в изречението, то някои от техните морфологични характеристики трябва да бъдат в определено систематично взаимоотношение (най-често, идентичност); ако това взаимоотношение не е спазено, то съответният случай на съгласуване е нарушен и поредицата е неграматична. (Разликата със селективните явления, тогава, се състои в това, че едновременната поява на двата (или повече) елемента Е1, Е2, … изобщо не е задължителна ─ т.е. съгласуването се нарушава, ако те се появят едновременно, без да са съгласувани, но не се нарушава, ако само един елемент от двойката (или множеството) се появи.)

Пример: поредицата **Мария щеше да му се е бил обадила.* нарушава съгласувателното отношение по род между двете части на сложната глаголна форма *бил(а) обадила* (докато изречението *Мария щеше да му се е обадила* е правилно ─ като тук трябва да се отбележи разликата в селективността).

Така изложеният преглед на класове явления ни води до предположението, че всяка поредица, която нарушава определено явление, може да бъде разглеждана като разширение на някаква ***минимална нарушаваща поредица***, т.е. като разширение на една поредица, която съдържа единствено материала, необходим за нарушаването. Например, ако неграматичната поредица *Директорът вчера бяха цялата вечер в къщи* бъде сметната за случай на нарушаване на съгласувателната връзка субект-обект, то тя може да бъде разглеждана като разширение на минималната поредица *Директорът бяха*.

Това означава, че една минимална нарушаваща поредица може да бъде открита във всяка неграматична поредица и следователно всяко «негативно правило» от (формалната) «негативна граматика» може да бъде съставено в две стъпки:

- първо, като се дефинира една (абстрактна) минимална нарушаваща поредица, базирана на нарушаването на едно конкретно явление;
- второ, като се дефинира как тази (абстрактна) минимална нарушаваща поредица може да бъде разширена до една напълно завършена (абстрактна) минимална нарушаваща поредица (или до повече такива поредици, ако съществуват повече възможности за разширение), т.е. чрез дефинирането на материала (по отношение на качество и позициониране), който може

да бъде добавен към минималната поредица, без да превърне новата поредица в неграматична.

И в двете стъпки трябва да се обърне внимание на това, че разглежданите поредици са с дължина на изречение[5] – т.е. те покриват понятието "неправилно изречение" от начало до край. За да стане това и формално ясно, специални знаци ще бъдат използвани в конфигурациите: '[' ще маркира началото на изречение (абстрактна позиция преди първата дума), ']' ще маркира край на изречение (т.е. абстрактна позиция "след крайната точка").

Пример: Абстрактната минимална нарушаваща поредица на примерната поредица *Директорът вчера бяха цялата вечер в къщи* има следната конфигурация (в обикновения израз на означаване, използващо признаковите структури на отделните елементи от обикновения израз и знакът '+' като разделител).

(1) $$[\ +\ \begin{bmatrix} \text{cat} : \text{n} \\ \text{gender} : \text{masc} \\ \text{num} : \text{sg} \\ \text{article} : \text{full} \end{bmatrix}\ +\ \begin{bmatrix} \text{cat} : \text{v} \\ \text{num} : \text{pl} \end{bmatrix}\ +\ ]$$

Тази конфигурация формулира, че поредица, състояща се от съществително в м.р. ед. ч. с пълен член, следвано от глагол в мн. ч. никога не може да бъде правилно изречение в български език.

След това една такава нарушаваща (абстрактна) поредица може да бъде генерализирана в неправилна конфигурация с неограничена дължина, както е описано по-долу, чрез комбиниране на лингвистични факти от българския синтаксис и тяхното инкорпориране в едно (полу)формално описание (което включва освен означението, описано в (1), също и звездата на Клиини '*' за всеки брой от повторни появи, и '!' за отрицание).

1. Добавяне на обстоятелствено пояснение към изречението не се отразява на субектно-обектното съгласуване в това изречение по никакъв начин, независимо дали това пояснение е наречие или предложна конструкция, с изключение на пояснението за придружаване (използващо предлога *"с"* като например *Директорът с жена си вчера бяха цялата вечер в къщи*). Това означава, че неопределен брой наречия, предлози и съществителни (но не и други части на речта!) могат да бъдат добавяни към поредица (1) без опасност тя да се превърне в граматична. Поредицата, която ще получи в резултат от това разширение, тогава може да бъде описана в (2):

---

[5] Това отразява вече споменатия факт, че само пълни поредици (т.е. не части от поредици) могат да бъдат смятани за граматични или неграматични в общия случай.

(2) $[+ \left([\text{cat} : \text{adj}] \vee \begin{bmatrix} \text{cat} : \text{prep} \\ \text{word} : \neg (\text{s} \vee \text{sys}) \end{bmatrix} \vee [\text{cat} : \text{n}] \right)^{\downarrow *}_{\uparrow} + \begin{bmatrix} \text{cat} : \text{n} \\ \text{gender} : \text{masc} \\ \text{num} : \text{sg} \\ \text{article} : \text{full} \end{bmatrix}$

$+ \left([\text{cat} : \text{adj}] \vee \begin{bmatrix} \text{cat} : \text{prep} \\ \text{word} : \neg (\text{s} \vee \text{sys}) \end{bmatrix} \vee [\text{cat} : \text{n}] \right)^{\downarrow *}_{\uparrow} + \begin{bmatrix} \text{cat} : \text{v} \\ \text{num} : \text{pl} \end{bmatrix}$

$+ \left([\text{cat} : \text{adj}] \vee \begin{bmatrix} \text{cat} : \text{prep} \\ \text{word} : \neg (\text{s} \vee \text{sys}) \end{bmatrix} \vee [\text{cat} : \text{n}] \right)^{\downarrow *}_{\uparrow} +]$

Следващото разглеждане за разширение на поредицата, без това да я направи – дори само вероятно – граматична взема предвид първо факта, че съществително от м.р. с пълен член може да бъде част от координация (като напр. *директорът и неговата жена*). Подобна координация по правило би се превърнала в субект в мн.ч., този субект тогава, от своя страна, взема глагол в мн.ч. като свой предикат. Следователно, ако неграматичността (нарушаването на субектно-глаголното съгласуване) трябва да се запази, такава координация не трябва да се появява. Това може да се подсигури чрез забраната за поява на съчинителен съюз в изречението. Съчинителните съюзи, обаче, могат да се появят навсякъде в изречението, което позволява допълнително намаляване на ограниченията (и следователно разширяване на множеството от описвани поредици) както следва.

(3) $[+ \left([\text{cat} : \text{adj}] \vee \begin{bmatrix} \text{cat} : \text{prep} \\ \text{word} : \neg (\text{s} \vee \text{sys}) \end{bmatrix} \vee [\text{cat} : \text{n}] \vee \begin{bmatrix} \text{cat} : \text{conj} \\ \text{type} : \text{subord} \end{bmatrix} \right)^{\downarrow *}_{\uparrow} + \begin{bmatrix} \text{cat} : \text{n} \\ \text{gender} : \text{masc} \\ \text{num} : \text{sg} \\ \text{article} : \text{full} \end{bmatrix}$

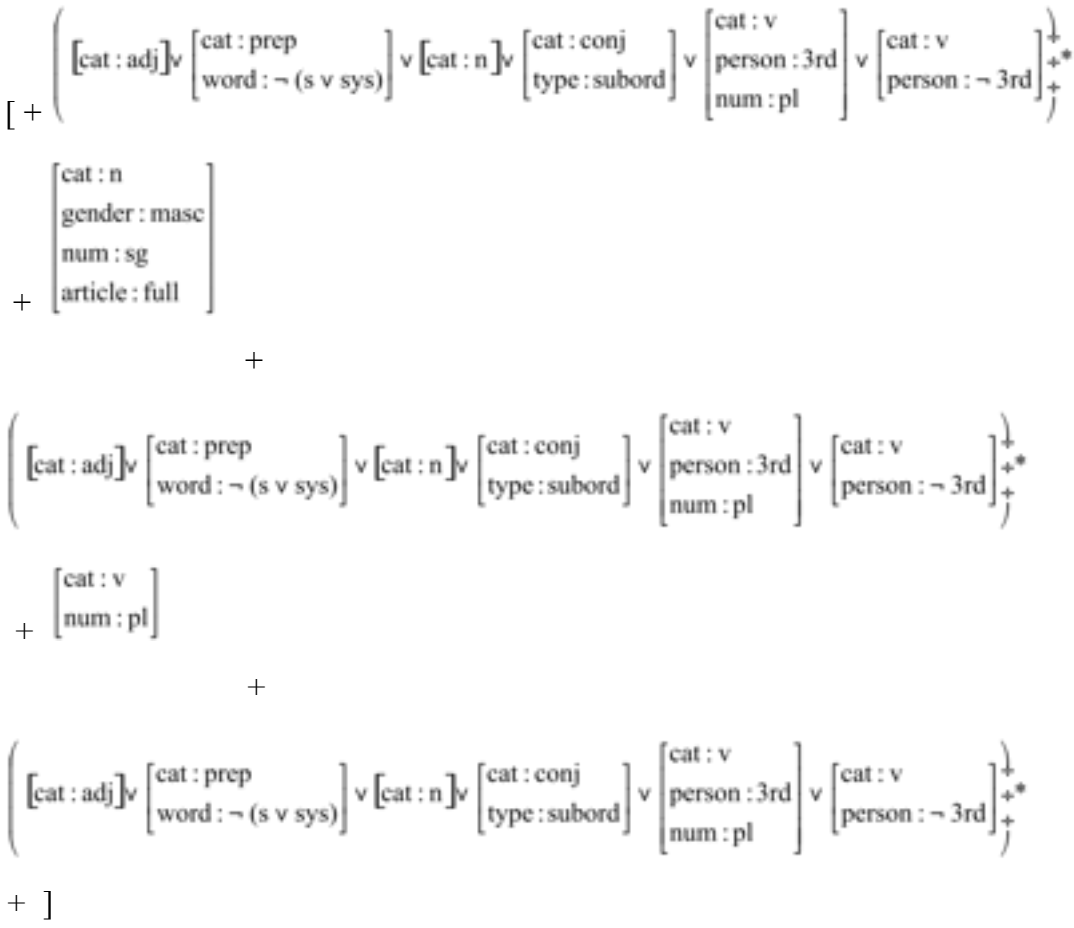
$+ \left([\text{cat} : \text{adj}] \vee \begin{bmatrix} \text{cat} : \text{prep} \\ \text{word} : \neg (\text{s} \vee \text{sys}) \end{bmatrix} \vee [\text{cat} : \text{n}] \vee \begin{bmatrix} \text{cat} : \text{conj} \\ \text{type} : \text{subord} \end{bmatrix} \right)^{\downarrow *}_{\uparrow} + \begin{bmatrix} \text{cat} : \text{v} \\ \text{num} : \text{pl} \end{bmatrix}$

$+ \left([\text{cat} : \text{adj}] \vee \begin{bmatrix} \text{cat} : \text{prep} \\ \text{word} : \neg (\text{s} \vee \text{sys}) \end{bmatrix} \vee [\text{cat} : \text{n}] \vee \begin{bmatrix} \text{cat} : \text{conj} \\ \text{type} : \text{subord} \end{bmatrix} \right)^{\downarrow *}_{\uparrow} +]$

За да остане разширяваната поредица неграматична, то трябва да бъдем сигурни, че нито един глагол в 3 л., ед.ч. няма да се появи в поредицата (тъй като ако такъв глагол се появи в поредицата, то той може да бъде сметнат за предиката на съществителното в ед.ч., докато глаголът в мн.ч. може да бъде разгледан като предикат, принадлежащ на друг субект). В частност това означава, че нито един глагол в 3 л. мн.ч. не трябва да се появява в цялото изречение (един пример на това – излишно – преправено изречение, т. е. на случая, който

трябва да се отхвърли, е *Директорът вчера замина, но другите бяха цялата вечер в къщи*). Това води до следното описание на разширение на неграматична поредица:

(4)



Тогава това, в настоящия пример, е окончателната форма[6] на описанието на едно абстрактно нарушаващо правило. Всяка конкретна поредица, отговаряща на това описание е гарантирано неграматична за български[7].

Като следваме вече описаното представяне на конструкцията на описанието на неграматична (абстрактна) поредица(и), е важно да се отбележи, че трябва да се процедира систематично при търсенето на такива поредици. Както вече стана ясно от предходната теоретична дискусия и от дадените примери, само лексикални категории (т. е. никакви фрази) могат да бъдат използвани за целите на дефинирането на формалната негативна граматика ─

---

6 Фактът, че това описание може да бъде допълнително уточнено, за да обхваща повече редици, ще бъде пренебрегнат в този пример.

7 От друга страна, ако една поредица не отговаря на това описание, очевидно нищо не може да бъде казано за нейната граматичност или неграматичност.

по начина вече представен по-горе. Причината за това е, че за неграматичните поредици се предполага, че нямат никаква йерархична (лингвистична) структура[8], което (както беше казано по-горе) е тяхна отличителна черта, и те трябва да бъдат разглеждани единствено като поредици от лексикални категории. Следователно ограниченията на формалната негативна граматика следва да бъдат абстракции на такива поредици, т.е. регулярни изрази от (абстрактни) лексикални категории. Съответно, "негативният език" ще бъде множество от всички поредици от думи, такива, че всяка от тези поредици да отговаря на поне един от тези регулярни изрази (където съпоставянето на индивидуалните категории е дефинирано като унификация, а съпоставянето на поредици по дължина и ред е дадено чрез регулярен израз).

По-нататък, както следва от вече изложените разсъждения, всяка от негативните конфигурации съдържа минимална нарушаваща поредица. Дължината на тези поредици варира в зависимост от случая.

Най-късата неграматична поредица в български може да бъде с дължина 1 (като границите на изречението [и] не се броят) и да изглежда, например, ето така:

(5) $[+ \begin{bmatrix} \text{cat} : \text{pron} \\ \text{form} : \text{clitic} \end{bmatrix} +]$

Тази неграматична поредица отразява факта, че местоименна клитика не може самостоятелно да образува изречение, тъй като тя се нуждае от някакъв материал, към който да се клитисизира (прикрепи). Тогава това нарежда (5) сред поредиците, които нарушават определено селективно явление, в случая такова, което изисква една местоименна клитика да се клитисизира (бъде прикрепена) към глагол или съществително име. В допълнение, това открива възможности за разширяване на минималната нарушаваща поредица (5) в разширени нарушаващи поредици (напр., чрез добавянето на каквото и да е количество материал, различен от глагол или съществително име).

Един пример за минимална нарушаваща поредица с дължина 2 беше даден в (1) и беше подробно дискутиран след това, включително и възможностите за разширяването му в разширена нарушаваща поредица. Очевидно, (1) е пример за нарушаването на съгласувателно явление.

А един пример за нарушаваща поредица с дължина 3 е предложен в (6):

---

8 Освен чисто линеарната подредба на техните елементи.

(6) $$[+\ [\text{cat}:\text{verb}]+\begin{bmatrix}\text{cat}:\text{pron}\\ \text{form}:\text{clitic}\\ \text{reflexivity}:-\\ \text{case}:\text{acc}\end{bmatrix}+\begin{bmatrix}\text{cat}:\text{pron}\\ \text{form}:\text{clitic}\\ \text{case}:\text{dat}\end{bmatrix}+]$$

Лингвистичният факт, стоящ в основата на този пример е, че ако винителна и дателна клитика се появяват заедно с глагол, то те трябва да стоят в реда винителна и дателна освен ако винителната клитика не е рефлексивна (с други думи, клитиката *се*). Обърнете внимание на неграматичността на *чете го му, чете го си*, но на граматичността на *чете му се*. Този случай е явно нарушение на словореда.

Въпреки че казаното дотук не може да се приеме като твърдение, че нарушенията при селектирането генерират минимални нарушаващи поредици с дължина 1, нарушенията при съгласуването генерират минимални нарушаващи поредици с дължина 2, а нарушенията на словореда пораждат минимални нарушаващи поредици с дължина 3, то все пак се внушава идеята, че търсенето на негативните правила на негативната граматика може да бъде извършено систематично като се използва категоризацията на явленията в комбинация с дължината на минималните нарушаващи поредици като ръководно начало, започвайки от поредици с дължина 1 и продължавайки във възходящ ред.

В предходните части бяха разгледани по-скоро теоретични въпроси, засягащи общия поглед върху понятието граматичност/неграматичност, както и начините за описание на граматични/неграматични поредици. Задачата за намирането на множеството от строго неграматични поредици, обаче, има също и практическо значение, тъй като за определени приложения в компютърната лингвистика е съществено важно да се знае, че конкретна конфигурация от думи (или от абстракции с множества от думи, напр., конфигурации с информация за части на речта) е гарантирано неправилна.

Най-известната (или поне: най-очевидната) сред тези задачи е (автоматичната) ***граматична проверка***: възможността за надеждно разпознаване на една поредица като неграматична ще доведе до ***граматични проверки*** със значително по-добро за потребителите представяне отколкото сега съществуващите (базирани предимно на прости техники за съпоставка на модели и следователно произвеждащи множество фалшиви аларми за правилни поредици от една страна, докато, от друга страна, оставят немаркирани много поредици, чиято неграматичност е очевидна за хората, но които не могат да бъдат разпознати като неправилни, тъй като тяхната вътрешна структура е прекалено комплексна (сложна) или не отговаря на никой от моделите по каквато и да е друга причина).

Друга практическа задача, където знанията за негативната граматика на определен език могат да се превърнат в основно необходимо познание е ***определянето /тагирането/ на частите на речта***, т.е. приписването на морфологична информация (като например част на речта, падеж, число, време, …) на думи в свързани текстове. Основният проблем за (автоматичното) определяне на частите на речта е морфологичната многозначност, т.е. фактът, че думите могат да имат различни морфологични значения (например, словоформата *работи* може да бъде съществително име (мн.ч. на *работа*) или глагол (както повелително наклонение, така и сегашно изявително или аорист). Знанието за неграматичните конфигурации може да бъде използвано за изграждането на тагер за частите на речта, базиран на идеята за (поетапно) елиминиране на тези конкретни /индивидуални/ четения, които са невъзможни в контекста на даденото изречение. В частност, всяка разширена нарушаваща поредица с *n* на брой съставящи я члена (т.е. конфигурация, която е образувана чрез разширяване на минимална нарушаваща поредица с дължина *n*) може да бъде превърната в множество от правила за разрешаване на многозначността чрез постановяване за всяко произтичащо правило на (*n*-1) съставни члена на разширената нарушаваща поредица като недвусмислени и задававане на израз за изтриване на *n*-тия оригинален елемент в поредицата, който съответства както на съставните елементи, така и на разширените елементи помежду им. По този начин всяка разширена нарушаваща поредица, породена от проста нарушаваща поредица с дължина *n,* поражда *n* на брой правила за разрешаване на многозначност.

Пример: Минималната нарушаваща поредица PREPOSITION/ПРЕДЛОГ + VERB/ГЛАГОЛ, след като е била разширена в конфигурацията (с обичайното означаване със звездата на Клийни) PREPOSITION/ПРЕДЛОГ + ADVERB/НАРЕЧИЕ* + VERB/ГЛАГОЛ, поражда следните две правила:

Правило 1: ***find_a_string*** (***намери_поредица***) съставена от (от ляво на дясно):

- дума, която е недвусмислен PREPOSITION/ПРЕДЛОГ (т.е. не носи друг таг/етикет или етикети освен PREPOSITION/ПРЕДЛОГ)
- произволен брой думи, които носят тага/етикета ADVERB/НАРЕЧИЕ (но никакви други етикети)
- дума, носеща тага/етикета VERB/ГЛАГОЛ

delete_the_ tag (***изтрий_етикета***) VERB/ГЛАГОЛ ***from (от)*** последната дума в поредицата

Правило 2: ***find_a_string*** (***намери_поредица***) съставена от (от ляво на дясно):

- дума, носеща тага/етикета PREPOSITION/ПРЕДЛОГ

- произволен брой думи, които носят тага/етикета ADVERB/НАРЕЧИЕ (но никакви други етикети)
- дума, която е недвусмислен VERB/ГЛАГОЛ (т.е. носи единствен етикет VERB/ГЛАГОЛ или носи повече от един етикет, но всички тези етикети са VERB/ГЛАГОЛ)

delete_the_ tag (***изтрий_етикета***) PREPOSITION/ПРЕДЛОГ ***from (от)*** първата дума в поредицата.

(Лингвистичната) валидност на тези правила се базира на факта, че която и да е поредица, съответстваща на частта от модела (*find_a_string*) от правилото на всяка позиция би била неграматична (в български), и следователно този прочит може да бъде изтрит (в *delete_the_ tag* частта от правилото) може да бъде премахнато, без негативни последици за някое от граматичните прочити/значения на изходната поредица.

Още една ─ но тясно свързана с другите две ─ задача, която може да се възползва до голяма степен от възможността за разпознаване на неграматичността е парсингът (синтактичният анализ). Ако информацията за частите на речта (т.е. морфологичната информация) на думите от входящата поредица може да се определи преди синтактичния анализ, или поне да се намали нейната морфологична многозначност, то тогава, очевидно, процесът на синтактичен анализ ще бъде чувствително по-ефективен. Така също, в случаите, когато за входящата поредица може със сигурност да се твърди, че е неграматична, нейният синтактичен анализ чрез обикновени методи може изобщо да не започва, като по този начин могат бъдат избегнати много процеси, отнемащи време (или като изцяло се откажем от синтактичния анализ, или като използваме някои техники за обработка на /справяне с/ неграматичните входящи поредици още от самото начало на синтактичния анализ).

2. СИСТЕМА ЗА ОПИСАНИЕ И ОБРАБОТКА НА ЛИНГВИСТИЧНИ ПРАВИЛА ParseEST

Системата за лингвистични правила ParseEst е базирана на крайни преобразуватели и автомати и предоставя средства за дефиниране, тестване и прилагане на правилата върху текст.

Модулът за дефиниране на лингвистични правила предоставя xml базиран език за описание на маркиращи правила. Правила от рода: маркирай А в контекст В и С, където А, В и С са регулярни изрази върху лингвистични единици и техните характеристики. Лингвистичните единици са дума и лема, а техните характеристики: част на речта и

граматични характеристики. Може да се опише и принадлежност на лингвистичните единици към даден лексикон (клас). XML-ът включва следните тагове:

{ ELEMENT GRAMMAR - (DEFINITION*, RULEGROUP+)

{ ELEMENT DEFINITION - (UNIFYNAME)

{ ELEMENT RULEGROUP - (PRIORITY, RULE+)

{ ELEMENT MARKRULE - (ANNOTATION?)

{ ELEMENT ANNOTATION - (MARK, LEFT?, RIGHT?)

{ ELEMENT MARK - (ELEMENT+, MATCH+, OR?, GROUP?, STAR?, PLUS?)

{ ELEMENT LEFT - (ELEMENT+, MATCH+, OR?, GROUP?, STAR?, PLUS?)

{ ELEMENT RIGHT - (ELEMENT+, MATCH+, OR?, GROUP?, STAR?, PLUS?)

{ ELEMENT ELEMENT - (pos?, posre?, npos?, nposre?, w?, nw?, l? nl?)

{ ELEMENT OR - (ELEMENT+, MATCH+)

{ ELEMENT GROUP - (ELEMENT+, MATCH+)

{ ELEMENT STAR - (ELEMENT, MATCH)

{ ELEMENT PLUS - (ELEMENT, MATCH)

Всяка граматика GRAMMAR трябва да съдържа поне една група правила RULE GROUP и може да има една или повече дефиниции DEFINITIONS. Всяка дефиниция DEFINITION има уникално множество от стойности на тагове, които могат да се унифицират. Всяка група правила RULE GROUP се състои от спецификация на приоритет за действие PRIORITY и едно или повече правила RULE. Приоритетът PRIORITY дефинира реда на обработка на групите правила RULE GROUPS. Всяко маркиращо правило MARKRULE се състои от анотация ANNOTATION, която съдържа задължителен елемент за маркиране MARK (означаващ съответстващата поредица) и може да има десен RIGHT и / или ляв LEFT контекст. Анотацията ANNOTATION е информацията, която се приписва на дадена поредица. Маркиране MARK, ляв контекст LEFT и десен контекст RIGHT се състоят от един или повече елементи ELEMENTS или техните абстрактни съответствия MATCHES и дефиниция на някой от следните оператори: или OR, група GROUP, звезда на Клийни STAR и плюс PLUS. Или OR и група GROUP оперират върху два или повече елементи ELEMENTS, или техни абстрактни съответствия MATCHES, докато звезда STAR и плюс PLUS - върху даден елемент ELEMENT или неговото съответствие MATCH. Елементът ELEMENT специфицира поредицата за разпознаване - дума, лема, част на речта и граматични характеристики, лексикон, не дума, не лема, не част на речта и граматични характеристики, регулярен израз върху част на речта и граматични характеристики.

Формализмът поддържа унификация на елемента част на речта и граматични характеристики в рамките на дадено правило. Правилата маркират части от текста на базата на контекстни или безконтекстни критерии. Под лексикон се разбира списък от думи, принадлежащи на един и същ клас: например географски понятия, собствени имена и т.н. Във формализма за описание на лингвистични правила думите от даден лексикон се цитират само с името на класа, към който принадлежат. Системата подържа работа с множество от различни лексикони, всеки един от които може да съдържа милиони думи, и дава възможност за дефиниране на обединение на класове. Специален модул, наречен компилатор на лексикони, преобразува списъците от думи в минимален ацикличен краен автомат с етикети в крайните състояния, които показват към кой клас (лексикон) принадлежи дадена дума (Михов 2000). Описанието на класовете, които са обединение на лексикони, се записва в отделен файл, а не в автомата. С помощта на автомата и на описанието на обединението на класовете може ефективно да се определи дали дадена дума от входния текст принадлежи към даден клас (лексикон).

Формализмът подържа каскадно прилагане на правилата, чрез групиране на правила в групи с предварително зададен приоритет. В рамките на една група правилата се прилагат в реда, в който са описани. Модулът за дефиниране/компилиране на правила преобразува xml описанието в двулентов краен автомат (преобразувател). Процесът по изграждане на крайния преобразувател протича по следния начин. За всяко правило от дадена група се конструира преобразувател, с помощта на библиотека за работа с крайни автомати (Foma). Foma (Хулден 2009) е разработена, за да бъде съпоставима по функционалност с Xerox toolkit (Бисли и Картунен 2003) по отношение на неговата експресивност и сила. Foma е среда със свободен код за конструиране и обработка на автомати и преобразуватели. Предоставя и C библиотека за интеграция на други модули; поддържа Уникод и различни формати за спецификация на регулярни изрази: Xerox/PARC формат, Perl-like формат и формат, който използва Mathematical Operators Unicode block (Хулден 2009). Като допълнение към базовите оператори за регулярни изрази е въведено използването на логика от първи ред (Хулден 2008).

Преобразувателите за правилата от дадена група се композират в реда, в който са описани правилата в рамките на групата. За всички правила от дадена група се получава един преобразувател, който съответства на последователното прилагане на правилата от групата. Така получените преобразуватели се композират в ред, отговарящ на зададения приоритет на групите от правила. По този начин на края се получава точно един преобразувател за всички

правила. Освен изходния преобразувател в резултат от компилирането на правилата се получава още един файл, който съдържа описанието на характеристиките на токъните, които участват в правилата (мета дефиниции), като и на кой символ от входната азбука на автомата отговаря тази комбинация от характеристики. Елементите представляват един символ от азбуката на преобразувателите. Ако имаме следния текст:

дума1 лема1 POS1 граматични характеристики1

дума2 лема2 POS2

дума3 лема3 POS3

дума4 лема4 POS4 граматични характеристики4

дума5 лема5 POS5

и следните дефиниции за мета символи:

замени дума1,,, с A

замени ,лема2,, с B

замени ,,POS3, с C

замени ,,,граматични характеристики4 с D

замени дума5 лема5 POS5 с E,

то разпознатата поредица ще бъде: ABCDE.

Прилагането на правилата протича в две стъпки. Първо входният текст се обработва от модул, наречен препроцесор. Ролята на препроцесора е да осъществи връзката между преобразувателя и формата на входния текст. Входният текст се разглежда токън по токън и се анализират характеристиките на токъните, анотирани от токънизатор, тагер и лематизатор - съответно токън, лема, част на речта и граматични характеристики. Прави се проверка дали дадена входна дума принадлежи към някой лексикон (посредством автомата на лексикона и описанието на класовете). След това се прави съпоставка с мета дефинициите (получени в резултат на компилирането на правилата). Ако има съпоставка между текущите характеристики на токъна и описанието на мета дефинициите, на входния токън се преписва символът от мета дефинициите. Ако входният токън не отговаря на нито една от мета дефинициите, му се преписва специален символ, който също е от входната азбука на автомата и означава че даденият токън не отговаря на нито една от мета дефинициите.

Така входният списък от анотирани токъни се представя като последователност от букви от входната азбука на автомата. Тази последователност се обработва от преобразувателя и се извличат символите, маркирани от него. Тъй като на всеки входен токън

отговаря точно един символ от входния стринг на преобразувателя, лесно са прави обратното съотношение между маркираните символи и входните токъни.

3. ПРОГРАМА ЗА ГРАМАТИЧНА КОРЕКЦИЯ WINEST+

Програмата за граматична корекция прилага правилата, дефинирани в системата ParseEst върху входен текст. Програмата за граматична корекция може да се разглежда като интегриран модул на всички програми за обработка на текста: токънизатор, разделител на изречения, тагер, лематизатор, парсер, като компонентите са комбинирани по максимално ефективен начин.

Ядрото на програмата за корекция на граматиката предоставя две функции:

- Намиране на първо изречение. Входният текст се обработва от програмата за разделяне на изречения и се връща позицията на първото изречение.
- Граматическа проверка. Входният текст се обработва последователно от токънизатора, тагера, лематизатора и модула за прилагана правилата. Извличат се индексите на токъните, маркирани като погрешни.

Програмата за проверка на граматиката е реализирана като 32-битова динамично-свързана библиотека (dynamic-linked library - DLL), която се зарежда от приложенията на MS Office. Избраният език за програмиране е C++. Връзката между приложенията на MS Office и Програмата за проверка на граматиката WinEst+ се осъществява посредством специален CGAPI (Common Grammar Application Interface) интерфейс, създаден от Майкрософт. Както повечето интерфейси на Майкрософт, така и този използва stdcall конвенция за извикване (при действие на функция аргументите влизат в стека в ред от дясно на ляво, а функцията сама “почиства” стека преди приключването си). Имената на всички функции са “недекорирани” - т.е. това са изцяло C функции.

Архитектурно всяка граматична проверка “живее” в своя собствена нишка (thread). Това налага употребата на някои синхронизиращи примитиви, за да бъдат избегнати конфликти при обработка на данните, които да доведат до грешни резултати или пък да предизвикат намеса от страна на операционната система и унищожаване на текущото приложение вследствие на неправилно изпълнени инструкции от страна на самата програма. Като синхронизиращ примитив е избрана критична секция (critical section). При MS Windows този примитив дава някои сериозни предимства пред употребата на стандартните mutex (MUtually EXclusive) или semaphore, когато става дума за междунишкова комуникация в рамките на един процес. Причината: стандартният mutex е обект от ядрото на операционната

система (kernel space), докато критичната секция е обект от адресното пространство на изпълнявания потребителски код (user space). При “заемането” на mutex всички извиквания водят до “прехвърлянето” им в ядрото на операционната система - т.е. за всяко извикване има проверка на аргументите на методите, търсене в структурите от данни на ядрото на операционната система и изпълняване на съответното действие. При заемането на критична секция, която преди това не е била заета, каквито са 90% от случаите, извикването е само в рамките на потребителското адресно пространство (user space) и не се налага “прехвърляне” на извикването в ядрото. Прехвърляне в ядрото на операционната система се налага единствено в случаите, когато критичната секция е заета от друга нишка и текущо изпълняваната нишка трябва да изчака, докато критичната секция не се освободи.

Основните компоненти на програмата за корекция на граматиката са: граматическо ядро (engine), диспечер на извикванията (call dispatcher) и граматиказатор (grammatizer). Граматическото ядро (engine) съдържа структури от данни и методи, необходими за проверка на граматиката. Поради своята идемпотентност то е подходящо за извиквания от различни нишки без особена нужда от синхронизация. Ето защо за всички сесии на всички приложения от пакета на MS Office това ядро има една инстанция, която се инициализира по време на първото зареждане на WinEst+ от дадено приложение. Всяко следващо зареждане на програмата за граматична проверка (било от нов документ в текущото приложение или от нов документ в ново приложение от пакета MS Office) увеличава с единица един “брояч на инстанциите” на ядрото. При приключване на сесия (затваряне на документ) този брояч се намалява с единица. При достигане на стойност нула на брояча ядрото се “самоунищожава” - т.е. освобождава заетата памет и работата приключва. От гледна точка на изпълнението на множество задачи едновременно местата, където е наложително да се ползват синхронизиращи примитиви, са там, където ядрото се създава (еднократно), и там, където се променя “броячът на инстанциите” (при отваряне или затваряне на документ на български език). За всички останали случаи достъпът до ядрото е директен - т.е. без ползване на синхронизиращи примитиви. По този начин се намалява обемът на общата заета памет за всички сесии на всички приложения от пакета MS Office.

Диспечерът на извикванията (call dispatcher) е основният “двигател” при работата с приложенията на MS Office. Съдържа в себе си таблица на регистрираните до момента обекти, обслужващи заявките по CGAPI интерфейса. Основен принцип при изграждането на CGAPI от страна на Майкрософт е идентификацията - всяко извикване на метод от CGAPI се идентифицира с номер - този номер е уникален в рамките на конкретно избрания модул (в

случая - модулът за проверка на граматика на български език). Тъй като е възможно много приложения от пакета на MS Office да работят с много сесии (отворени документи), голяма част от които - на български език - е необходимо всяко извикване за проверка на граматиката да бъде лесно идентифицирано - т.е. да се знае кой документ с кой реален обект за проверка на граматиката (граматиказатор) е свързан. Въпреки че ядрото за проверка на граматиката е идемпотентно от гледна точка на методите, които предлага, това не се отнася до граматиказатора и това налага необходимостта от обект като диспечер на извикванията. От гледна точка на изпълнението на множество от задачи диспечерът на извикванията също е защитен от синхронизиращия примитив (неговата таблица). При всеки достъп (четене, добавяне, изтриване) на елемент от таблицата се минава през синхронизиращия примитив. Както при ядрото, така и при диспечера - инициализацията става по време на първото зареждане на програмата, а освобождаването на паметта - при затварянето на последната сесия (документ). Друга основна задача на диспечера на извикванията е да определи текущата версия на MS Office и в зависимост от нея да създаде подходящ граматиказатор.

Ако ядрото (engine) е “сърцето” на програмата за проверка на граматиката, то граматиказаторът е “мозъкът” - той управлява ядрото и решава кога и какви действия да се предприемат в зависимост от входния текст и вътрешните си състояния. Всеки граматиказатор има собствен уникален идентификатор - по този идентификатор диспечерът на извикванията знае за кой граматиказатор е предназначено извикването от страна на текущото приложение. Граматиказаторът не е идемпотентен от гледна точка на методите си, но благодарение на диспечера на извикванията е защитен от “състезания” (race conditions).

Всяко приложение от пакета MS Office, което използва програмата за граматична проверка, първо определя входния си език. Без подходящ входен език не е възможна проверка на граматиката. При “откриване” на входния език приложението зарежда вече регистрирания в системата модул на програмата за проверка на граматиката за съответния език. Това може да стане в зависимост от настройките на приложението - по време на писане (докато потребителят пише) или изрично по негова (на потребителя) заявка. По време на първоначалното зареждане се инициализират ядрото и диспечера на извикванията и се определя версията на инсталирания MS Office пакет. След като се зареди модулът на програмата за проверка на граматиката, приложението проверява версията на подържания CGAPI интерфейс от програмата за проверка на граматиката за някои “характеристики”. Например - може ли да работи във фонов режим, може ли да определя границите на изреченията, нужен ли е модул за проверка на правописа. Следва извикване на

приложението, което създава граматиказатор на база текущо инсталираната версия на MS Office, регистрира този граматиказатор в таблицата на диспечера на извикванията и връща идентификатора на граматиказатора на приложението. След това всички извиквания се регистрират посредством този идентификатор.

Ако програмата за граматична проверка може да работи във фонов режим, извикванията за проверка на входния текст са напълно асинхронни и непрозрачни за потребителя - т.е. те остават скрити, без видимо забавяне на потребителския интерфейс. При проверка на правописа приложението от MS Office пакета подава входния текст и изисква от програмата за граматична проверка анализ само на първото изречение. Този анализ може да включва определяне на границите на изречението, проверка на правописа на думите или граматическа проверка. Програмата за граматична проверка връща информация за началото и края на първото изречение и наличието на грешки - правописни, граматични, други (например непознат език). При наличие на грешка програмата за граматична проверка предоставя информация за всяка грешка - позиция и дължина в изречението, обяснение (на конкретния език) на грешката, възможности за корекция, предложения за корекция. След върната информация от програмата за граматична проверка приложението маркира неправилния текст в зелено и при по-нататъшни действия на потребителя (десен бутон върху зеленото подчертаване) предлага диалог, в който подробно показва възможностите за корекция, обяснение на грешката, тип на грешката и т.н.

При разработката на програмата за граматична проверка WinEst+ са взети под внимание следните особености: като платформа за разработване е избрана Windows 7 (като сравнително нова и най-разпространена), MS Office 2007 (най-разпространен пакет) и 32-битова версия на ядрото на Windows 7. Майкрософт препоръчва за крайни потребители 32-битови версии на офис продуктите си поради зависимост от добавките на други производители.

Литература:

Бисли и Картунен 2003: Beesley, K. and L. Karttunen: Finite-State Morphology. CSLI, Stanford. (2003)

Карлсън и др. 1995: Karlsson F., A. Voutilainen, J. Heikikilä, and A. Antilla (eds.): Constraint Grammar ─ A Language-Independent System for Parsing Unrestricted Text. Mouton de Gruyter, Berlin & New York 1995

Келър 2000: Keller F. Gradience in Grammar: Experimental and Computational Aspects of Degrees of Grammaticality. PhD Thesis, University of Edinburgh 2000

Холън и др. 2003: Holan T., V. Kuboň, K. Oliva and M Plátek: A Theoretical Basis of an Architecture of a Shell of a Reasonably Robust Syntactic Analyser, in: Proceedings of the conference Text, Speech and Dialogue 2003, Lecture Notes in Artificial Intelligence, Springer 2003

Хулден 2008: Hulden, M.: Regular expressions and predicate logic in finite-state language processing. In: Proceedings of FSMNLP 2008. eds., Piskorski, J., Watson, B., and Yli-Jyra, A.

Хулден 2009: Hulden, M.: Foma: a finite-state compiler and library. In: Proceedings of the EACL Demonstration Session, pages 2932. (2009)