МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ
ІМЕНІ АДМІРАЛА МАКАРОВА

Димо О.Б.

КОМП'ЮТЕРНА ЛІНГВІСТИКА

ЧАСТИНА 1: СИНТАКСИЧНИЙ АНАЛІЗ ПРИРОДНИХ МОВ

Навчально-методичний посібник з курсу
“Лінгвістичне забезпечення комп'ютерних систем”
для студентів спеціальності 7.030.505
“Прикладна лінгвістика”

Миколаїв, 2010

Зміст

ВСТУП

Комп'ютери вже давно стали вагомою складовою сучасної людської цивілізації. Їх створили як інструмент для швидкого вирішення прикладних задач фізико-математичних розрахунків, але дуже скоро люди усвідомили як близько вони підійшли до здійснення своєї давньої мрії — створення інтелекту, рівного людському.

Природно, що такий інтелект (якщо він буде створений) має спілкуватися з нами нашою же мовою. Досі немає такого комп'ютера і такої програми, які би розуміли і вміли вільно користуватися природними мовами. Тим не менш, за останні 50 років ми значно наблизились до вирішення такої задачі.

Сьогодні комп'ютер є основним засобом комунікацій між людьми, середовищем, в якому виконується мовна взаємодія. Текстові редактори розпізнають і коригують помилки правопису, граматичні і стилістичні помилки в текстах. Програми розпізнають звуки людської мови і перетворюють їх в текст, перекладають з однієї мови на іншу, шукають інформацію в текстах і дають відповіді на запитання. Автоматизовані діалогові системи виконують функції автовідповідачів, помічників і діагностів.

Все це стає можливим завдяки програмам, які реалізують відомі нам технології обробки природних мов на комп'ютерах. Цей посібник має перед собою мету познайомити з такими технологіями, їх можливостями і обмеженнями, дати приклади використання і вказати шляхи подальшого їх розвитку.

Будь-які комп'ютерні технології складаються з двох складових частин — із формальної логіко-математичної моделі явища, яке відтворює його на комп'ютері і дозволяє запрограмувати, і з способів

або алгоритмів вирішення конкретних прикладних задач. Тому і посібник складається з двох частин. В першій викладені формальні моделі природної мови, а в другій представлені найцікавіші способи і алгоритми обробки текстів на комп'ютері.

Посібник розрахований насамперед на студентів лінгвістичних і філологічних напрямків, тому в ньому не розглядаються питання програмування систем обробки природних мов, а супровідна математика викладена в основному неформально і через приклади.

В основу посібника покладений лекційний курс “Лінгвістичне забезпечення комп'ютерних систем”, що викладався на протязі 2006-2010 рр. в Національному університеті кораблебудування імені адмірала Макарова студентам спеціальності “Прикладна лінгвістика”. Матеріал, викладений в посібнику розраховано на два семестри навчання.

Автор висловлює подяку професору Філіповій Н.М. за можливість на протязі чотирьох років викладати цей курс студентам університету.

1. МОВНА КОМПЕТЕНЦІЯ КОМП'ЮТЕРІВ. ФОРМАЛЬНІ МОВИ ТА ГРАМАТИКИ

Кожна людина при народженні отримує засоби, які дозволяють їй користуватися мовою — мовну компетенцію за Хомським. Людина реалізує свою мовну компетенцію коли вивчає мову і здобуває можливість сприймати, оброблювати і передавати інформацію.

Якщо ми бажаємо задіяти можливості комп'ютерів для сприйняття, обробки і передачі інформації (так як це робить людина), ми повинні закласти в них механізм забезпечення мовної компетенції. Такий механізм має вирішувати три задачі:

1. задачу розпізнавання мови не залежно від інструменту її передачі — голосу або письмового тексту;
2. задачу визначення смислу і значення отриманої інформації, її обробки;
3. задачу генерації мови для передачі інформації.

Сучасні комп'ютери працюють за законами математики та логіки, тому реалізація механізму мовної компетенції потребує відтворення мовних явищ в комп'ютері — комп'ютерної математичної моделі мови. Для моделювання потрібно визначити мову як формальний (математичний) об'єкт.

Лінгвісти наводять декілька визначень мови:

1. Сепір: мова є чисто людський, неінстинктивний спосіб передачі думок, емоцій та бажань за допомогою системи спеціальних символів;
2. Блок і Трегер: мова — це система довільних голосових символів, за допомогою якої виконується взаємодія в деякій соціальній групі;
3. Холл: мова — це інститут, який забезпечує взаємодію людей за

допомогою довільних символів, що використовуються за звичкою;

4. **Хомський: мова — це множина (скінченна або нескінченна) речень, кожне з яких має скінченну довжину і побудовано зі скінченної множини елементів**.

Перше та третє визначення стверджують що мова притаманна тільки людині. Друге та третє — що мова є системою довільних символів, які сучасні комп'ютери не здатні оброблювати завдяки відсутності закономірностей. Тому мова за визначеннями Сепіра, Блока, Трегера і Холла не може мати комп'ютерної моделі.

Хомський визначив мову в термінах множин речень і складових елементів. Множина є формальним математичним поняттям, тому визначення Хомського є найкраще придатним для комп'ютерного моделювання мови.

Також на відміну від інших, Хомський не визначив мову як суто людське явище. Лінгвісти часто вважають це побічним ефектом формального визначення, але для комп'ютерної обробки мов побічний ефект виявився важливим. Мова взаємодії людини з технікою (не тільки комп'ютерною), комп'ютерні мови програмування — все це відповідає визначенню мови Хомського. Багато лінгвістів не згодні з таким визначенням мови, тому аби запобігти двозначності, надалі ми будемо називати всі мови, які підпадають під визначення Хомського, формальними.

Визначення: Математично формальна мова L визначається її граматикою $G(L)$ як сукупність з чотирьох елементів:

$$G(L) = (T, N, S, P),$$

де T — множина термінальних символів, або “алфавіт” мови;
N — множина нетермінальних символів, або синтаксичних категорій, з яких формуються складові речень мови;

S — кореневий символ, найбільш абстрактна синтаксична категорія, з якої формуються повні речення мови;

P — множина продукцій, або правил граматики, які визначають як будуються речення мови (або більш формально, як нетермінали перетворюються в рядки з терміналів).

Продукції описують правила перетворення рядків з нетерміналів і терміналів і мають вигляд $\alpha \rightarrow \beta$, де α і β — рядки з нетерміналів і терміналів.

Речення мови повинні породжуватись з кореневого символу *S* шляхом заміни його на рядки з терміналів і нетерміналів за правилами заміни *P* до того часу, коли в рядку не залишаться тільки термінали. На цьому породження завершується, а отриманий рядок стає реченням мови.

Приклад 1.1: Для пояснення визначення формальної мови, побудуємо граматику мови, яка складається з одного речення

кішка спіймала мишку

Згідно з визначенням, терміналами мови будуть символи, з яких складається речення. Наше речення складається з трьох слів — *кішка*, *спіймала*, *мишку*, тобто вони й будуть терміналами мови або її алфавітом:

$$T = \{ \textit{кішка}, \textit{спіймала}, \textit{мишку} \}.$$

Увага, визначення алфавіту формальної мови відрізняється від визначення алфавіту природної мови! В формальних мовах слово не розподіляється на окремі букви, тому й алфавіт мови складається не з букв, а зі слів.

Кореневим символом, з якого формується (породжується) речення буде саме “речення”, бо воно і є найбільш абстрактною синтаксичною категорією для речень мови. Надалі кореневий символ ми будемо позначати літерою *S* (від слова Sentence). Всі формальні граматики, які

моделюють природні мови будуть мати *S* кореневим символом.

Нетермінали (синтаксичні категорії) формальної мови визначимо за аналогією з українською мовою. В українській мові речення *“кішка спіймала мишку”* складається з фрази іменника і фрази дієслова. Фраза іменника складається з одного іменника *“кішка”*. Фраза дієслова — з дієслова *“спіймала”* і зі фрази іменника *“мишку”* — додатку до дієслова.

Позначимо іменник символом *N* (Noun), дієслово — символом *V* (Verb), фразу іменника символом *NP* (Noun Phrase) і фразу дієслова символом *VP* (Verb Phrase). Ці символи і символ кореневого слова *S* разом складають множину нетерміналів

$$N = \{ S, NP, VP, N, V \}.$$

Правила граматики, що були неформально визначені вище, запишемо відповідно до вимог написання продукцій формальної граматики:

“речення складається з фрази іменника і фрази дієслова”: $S \to NP\ \ VP$

“фраза іменника складається з одного іменника”: $NP \to N$

“фраза дієслова — з дієслова і зі фрази іменника”: $VP \to V\ \ NP$

Якщо додати до цих правил, визначення конкретних іменників і дієслів, ми отримаємо множину продукцій *P*:

$S \to NP\ \ VP$

$NP \to N$

$VP \to V\ \ NP$

$N \to кішка \mid мишку$

$V \to спіймала$

Тут і надалі правило виду “$N \to кішка \mid мишку$” еквівалентно двом правилам “$N \to кішка$” та “$N \to мишку$”. Знак “|” можна назвати словом “або”, тоді все правило має значення “іменник — це або *кішка* або *мишку*”.

Вся граматика мови записується як:

G(простого речення української мови) = (T, N, S, P)

T = { *кішка*, *спіймала*, *мишку* }

N = { S, NP, VP, N, V }

$S = S$

P = { $S \rightarrow NP\ \ VP$

$NP \rightarrow N$

$VP \rightarrow V\ \ NP$

$N \rightarrow$ *кішка* | *мишку*

$V \rightarrow$ *спіймала* }

Правильно записана граматика повинна породжувати вихідне речення. Породження починається завжди з кореневого символу S. Граматика визначає тільки одне правило для заміни S на фразу іменника і фразу дієслова:

$$S => NP\ \ VP$$

Увага, при запису породження використовується знак подвійної стрілки "=>" аби відрізнити його від знаку стрілки "→" в правилах. Знак "→" означає можливість заміни, а знак "=>" — саму заміну.

Так як рядок "$NP\ \ VP$" складається з нетерміналів, продовжуємо породження, використовуючи правила граматики для заміни NP і VP. Наприкінці замінюємо нетермінали іменників і дієслова на конкретні іменники і дієслово:

$$S => NP\ \ VP => N\ \ VP => N\ \ V\ \ NP =>$$
$$=> N\ \ V\ \ N =>$$ *кішка спіймала мишку*.

<u>Приклад 1.2:</u> Завдяки своєму визначенню, формальні мови описують не тільки природні мови, а й штучні, такі як мова арифметичних операцій або проста мова програмування. Наприклад, побудуємо граматику мови виразів виду

$$2 + 2 / 6$$

Терміналами мови будуть цифри і знаки додавання, віднімання,

добутку і ділення:

$$T = \{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, +, -, /, \cdot \}.$$

Нетерміналами будуть складові операції арифметичного виразу — додавання, множення і цифри. Все "речення" мови арифметичних операцій складається з додавань, тому додавання і буде кореневим символом *S (Sentence або Sum)*. Множення позначимо як *M* (Multiplication), а цифри — як *D* (Digit). Тоді множина нетерміналів мови буде:

$$N = \{ S, M, D\}.$$

Правила мови арифметичних операцій повинні відображати пріоритет операцій — спочатку виконується множення (або ділення), а потім додавання (або віднімання), тобто:

"множення — це множення цифр або цифра": $M \rightarrow D / D \mid D \cdot D \mid D$

"додавання—це додавання результатів множення": $S \rightarrow M + M \mid M - M$

Повністю, граматика мови арифметичних операцій записується як:

G(мови арифметичних виразів) = (T, N, S, P)

$T = \{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, +, -, /, \cdot \}$

$N = \{ S, M, D\}$ $S = S$

$P = \{ S \rightarrow M + M \mid M - M$

$M \rightarrow D / D \mid D \cdot D \mid D$

$D \rightarrow 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \mid 0 \ \}$

Правильність граматики доводить породження вихідного виразу:

$$S => M + M => D + D / D => 2 + 2 / 6.$$

Приклад 1.3: Не тільки мови програмування є формальними мовами. Всі інші засоби комунікації людини з технікою належать до класу формальних мов, що можна проілюструвати на прикладі мови комунікації глядача і його телевізора.

Типовий сценарій комунікації глядача і телевізора наступний: аби

подивитися телевізор, людина бере в руки пульт керування і натискає кнопку включення. Потім вона вибирає один за одним канал і дивиться його. Наприкінці, людина натискає кнопку виключення.

Авжеж, наведений сценарій дещо спрощений, але він відтворює можливу в реальності послідовність дій людини. Позначимо дію зі включення символом "*on*", дію з виключення — символом "*off*", а дії з перемикання каналів — номерами каналів (наприклад, від "1" до "9"). Тоді можливою послідовністю дій глядача буде така:

on 1 5 1 2 *off*

Записана символами послідовність дій глядача одразу нагадує речення якоїсь мови. І дійсно, можна написати формальну граматику такої мови:

$G = (T, N, S, P)$

$T = \{ on, off, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$

$N = \{ S, Channels, Channel \}$

$P = \{ S \rightarrow on\ Channels\ off$

$Channels \rightarrow Channel\ Channels \mid Channel$

$Channel \rightarrow 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \}$

Ця граматика має дві важливі особливості. По-перше, з неї породжуються правильні речення мови комунікації глядача і телевізора:

$S \Rightarrow on\ Channels\ off \Rightarrow on\ Channel\ Channels\ off \Rightarrow$

$\Rightarrow on\ 1\ Channels\ off \Rightarrow on\ 1\ Channel\ Channels\ off \Rightarrow$

$\Rightarrow on\ 1\ 5\ Channels\ off \Rightarrow on\ 1\ 5\ Channel\ off \Rightarrow on\ 1\ 5\ 1\ off$.

По-друге, граматика є рекурсивною. Зверніть увагу на правило для нетерміналу *Channels*. Смисл цього правила в тому, що глядач може переглядати послідовно довільну кількість каналів — перемикати телевізор на інший, повертатися до попереднього і таке інше. В формальній граматиці довільна послідовність реалізується рекурсивним правилом, в якому з нетерміналу породжується рядок

символів, що включає в себе сам цей нетермінал.

Правило "*Channels → Channel Channels*" є прикладом рекурсивного правилу. Воно буквально означає, що список каналів, які дивиться глядач складається із одного каналу і продовження цього списку. Це правило можна застосовувати скільки завгодно довго (замінюючи нетермінал *Channels* на будь-який канал і самого себе), тому до нього додається нерекурсивне правило для закінчення породження виду "*Channels → Channel*".

Наприклад, для породження рядку з двох каналів правило слід застосувати один раз рекурсивне правило:

Channels => Channel Channels

і один раз нерекурсивне правило для закінчення породження:

Channel Channels => Channel Channel .

Для породження рядку з трьох каналів слід застосувати рекурсивне правило два рази:

Channels => Channel Channels => Channel Channel Channels =>

=> Channel Channel Channel .

Приклад 1.4. Граматики природних мов теж мають рекурсивні правила. Наприклад, перед іменником в реченні можуть бути написані прикметники, які його характеризують:

сірий пухнатий довгий хвіст

Зверніть увагу, що таких прикметників може не бути взагалі або бути як завгодно багато. Їх кількість обмежена тільки можливостями сприйняття людини, а не граматикою української мови.

Подібні речення можуть бути породжені з правил

$$S \rightarrow Adjectives\ Noun$$

$$Adjectives \rightarrow Adjective\ Adjectives \mid \varepsilon$$

Останнє правило дозволяє породжувати довільну кількість прикметників або не породжувати жодного. Правило "*Adjectives* → ε"

є правилом породження пустого рядка, який позначається літерою ε. Такі правила дозволяють в будь-який момент породження припиняти застосування рекурсивного правилу (так само як і правило "*Channels → Channel*" в прикладі 1.3), та на додаток вони дозволяють не породжувати нічого (тобто, породжувати тільки іменник з речення).

Висновки. Наведені вище приклади демонструють, що:

- формальні мови дозволяють моделювати природні мови (приклади 1.1 і 1.4, в яких моделюється українська мова);
- формальні мови можуть визначати мови як зі скінченною кількістю речень (приклади 1.1 і 1.2), так і, завдяки рекурсивним правилам, з нескінченною кількістю речень (приклади 1.3 і 1.4);
- формальні мови мають такі ж виразні можливості, як і природні мови — вони здатні моделювати граматичні явища природних мов і визначати мови з нескінченною кількістю речень так само як і природні мови;
- формальні мови можуть бути моделями не тільки природних мов, а і мов штучних (програмування, математики, комунікацій людини і її техніки).

Ці властивості формальних мов дозволяють реалізовувати їх в комп'ютерних програмах — механізмах мовної компетенції комп'ютерів. З трьох задач цих механізмів (розпізнавання, обробка і генерація мови), формальні мови вирішують як мінімум дві. Генерація речень мови — це породження нових речень із кореневого символу з використанням правил формальної граматики. Розпізнавання речень мови — це задача реконструкції граматики по реченню з використанням правил формальної граматики в зворотньому порядку.

Тільки два питання залишаються нерозглянутими. По-перше, як добре формальна мова моделює всі граматичні явища природної (такі як відмінки іменників, дієвідміни дієслів, рід, число та інші). А по-

друге, як добре формальні мови відтворюють смисл, що несуть в собі речення і як вони вирішують семантичні і прагматичні проблеми мови (такі як узгодженість об'єкту і суб'єкту, об'єкту і його характеристик, дозволеність дії та інші).

В наступних підрозділах буде розглянуто які є типи формальних мов і як добре вони відтворюють природні мови.

2. КОНТЕКСТНО-ВІЛЬНІ МОВИ І ГРАМАТИКИ

Всі приклади формальних мов, які розглянуті в попередньому розділі, мають спільну особливість - правила граматики мають вигляд "$A \rightarrow \alpha$", де A — нетермінал, а α — рядок з терміналів і нетерміналів. Граматики з такими правилами називаються **<u>контекстно-вільними</u>**. Так само й мови, які вони визначають, є контекстно-вільними.

Назву "контекстно-вільна" пояснює приклад 1.1. граматики мови з одного українського речення "*кішка спіймала мишку*". Породження, яке було проведено з використанням граматики записується як:

$$S => NP\ \ VP => N\ \ VP => N\ \ V\ \ NP =>$$
$$=> N\ \ V\ \ N => кішка\ спіймала\ мишку.$$

Але це породження є не єдиним дозволеним граматикою. Нетермінал іменника N дозволено змінювати на будь-який іменник — чи то "*кішка*" чи то "*мишку*". Тому насправді граматика визначає мову з чотирьох речень:

1) $S => NP\ \ VP => N\ \ V\ \ N => кішка\ спіймала\ мишку$
2) $S => NP\ \ VP => N\ \ V\ \ N => кішка\ спіймала\ кішка$
3) $S => NP\ \ VP => N\ \ V\ \ N => мишку\ спіймала\ мишку$
4) $S => NP\ \ VP => N\ \ V\ \ N => мишку\ спіймала\ кішка$

З цих чотирьох речень тільки два є правильними з точки зору української мови, тобто ми спостерігаємо неадекватність моделі мови (яка визначається формальною граматикою) самій мові.

Тут неадекватність моделі є в тому, що граматика не враховує

правила відмінювання іменників (синтаксичні правила), дозволяючи речення "*кішка спіймала кішка*" і "*мишку спіймала мишку*". Це відбувається тому, що іменники в знахідному відмінку можуть використовуватися на місці іменників в називному відмінку і навпаки. Правила не враховують, що на місці підмету дозволено тільки іменник в називному відмінку, а на місці додатку — тільки іменник в знахідному. Тобто, заміна нетерміналу *N* виконується незалежно від місця, в якому ця заміна відбувається, або іншими словами, у відриві від контексту. Тому формальні граматики, що принципово дозволяють такі заміни, називають контекстно-вільними.

Можливість породження неправильних речень контекстно-вільною граматикою не заважає створювати такі граматики, що породжують тільки синтаксично правильні речення.

Приклад 1.5. Правила заміни іменників з граматики прикладу 1.1. можна уточнити, вказавши для кожного іменника його відмінок, наприклад як:

$S \rightarrow NP\ \ VP$

$NP \rightarrow Nn$

$VP \rightarrow V\ Na$

$V \rightarrow$ *спіймала*

$Nn \rightarrow$ *кішка* | *мишка*

$Na \rightarrow$ *кішку* | *мишку*

Останні два правила з нетерміналами *Nn* (Noun, nominative) і *Na* (Noun, accusative) є достатнім уточненням аби граматика визначала мову чотирьох синтаксично правильних речень української мови:

1) $S \Rightarrow NP\ \ VP \Rightarrow Nn\ \ V\ \ Na \Rightarrow$ *кішка спіймала мишку*
2) $S \Rightarrow NP\ \ VP \Rightarrow Nn\ \ V\ \ Na \Rightarrow$ *кішка спіймала кішку*
3) $S \Rightarrow NP\ \ VP \Rightarrow Nn\ \ V\ \ Na \Rightarrow$ *мишка спіймала мишку*
4) $S \Rightarrow NP\ \ VP \Rightarrow Nn\ \ V\ \ Na \Rightarrow$ *мишка спіймала кішку*

Хоча така уточнена граматика визначає правильні речення з точки зору синтаксису, друге, третє і четверте речення не мають смислу в контексті реального миру. В реченнях, в яких суб'єкт ловить об'єкт, на місцях об'єкту і суб'єкту можуть виступати тільки певні іменники, тому граматика потребує ще одного уточнення:

$S \rightarrow NP\ \ VP$

$NP \rightarrow Nns$

$VP \rightarrow V\ Nao$

$V \rightarrow$ *спіймала*

$Nns \rightarrow$ *кішка*

$Nno \rightarrow$ *мишка*

$Nas \rightarrow$ *кішку*

$Nao \rightarrow$ *мишку*

де нетермінал *Nns* позначає іменник (N) в називному відмінку (n), який виступає суб'єктом (s); *Nno* — іменник (N) в називному відмінку (n), який виступає об'єктом (o); аналогічно *Nas* і *Nao* — іменники суб'єкту і об'єкту в знахідному відмінку. Граматика тепер породжує тільки одне речення, правильне з точки зору і синтаксису і семантики:

$S \Rightarrow NP\ \ VP \Rightarrow Nns\ \ V\ \ Nao \Rightarrow$ *кішка спіймала мишку* .

Висновки. Контекстно-вільні граматики і мови здатні моделювати різноманітні синтаксичні і семантичні явища природних мов. Приклади показують, що уточнення граматики шляхом доповнення її новими нетерміналами і правилами є основним засобом відображення правил синтаксису і семантики в контекстно-вільних мовах.

В цьому є водночас основна перевага і основний недолік контекстно-вільних мов як засобів моделювання природних мов. Добавити в граматику нові синтаксичні категорії і правила просто (це перевага), але легко спрогнозувати, що для повного відображення всіх

явищ природної мови потрібна буде велика кількість правил (це недолік).

Для штучних мов контекстно-вільні граматики є найкращим відомим засобом комп'ютерного представлення. Всі існуючі мови програмування є або контекстно-вільними, або можуть бути з деякою точністю розпізнаними контекстно-вільною граматикою.

Важливим також є те, що комп'ютерні алгоритми розпізнавання і генерації контекстно-вільних мов є добре відомими, а програми, що їх реалізують мають добру продуктивність, тобто швидко виконуються на сучасних комп'ютерах.

На сьогодні всі комп'ютерні програми обробки природної мови, які основані на формальних мовах, використовують контекстно-вільні граматики. Наприклад, перевірка граматики в текстових редакторах реалізована саме таким чином.

Отже, контекстно-вільні мови є адекватними моделями природних мов, добре пристосованими для комп'ютерної обробки, але трудомісткими для розробки.

Частково трудомісткість пояснюється великою кількістю правил, частково самим процесом розробки граматики мови, який можна умовно назвати як розробка “зверху-вниз” — від правил до конкретних речень. В наступних підрозділах буде розглянуто як, по-перше, зменшити кількість правил граматики, а, по-друге, як можна побудувати граматику знизу-наверх, тобто на основі речень вивести правила мови.

3. КОНТЕКСТНО-ЗАЛЕЖНІ ГРАМАТИКИ

Негативного впливу незалежності від контексту, як це було показано в попередньому підрозділі, можна уникнути за допомогою доповнення граматики новими правилами. Альтернативою цьому може бути введення самого поняття “контекст” в граматику і дозвіл

замінювати нетермінали в контексті.

Граматика, в якій правила мають вигляд "$\xi_1 A \xi_2 \rightarrow \xi_3 \alpha \xi_4$", де A — нетермінал, α — рядок з терміналів і нетерміналів, ξ_1, ξ_2, ξ_3, ξ_4 — контексти з терміналів і нетерміналів, називається **контекстно-залежною** граматикою. В таких граматиках заміна нетерміналу A на рядок α проводиться в контексті його оточення ξ_1 і ξ_2.

Приклад 1.6. Запишемо контекстно-залежну граматику для мови речення "*кішка спіймала мишку*". Вище було сказано, що заміна нетерміналу "іменник" повинна відбуватися в залежності від контексту. В цьому реченні є два контексти — контекст суб'єкту (позначимо його літерою σ) і контекст об'єкту (позначимо його як ó). Тоді правило заміни іменника в контексті суб'єкту на слово "*кішка*" можна записати як "N σ → *кішка*", а правило заміни іменника в контексті об'єкту як "N ó → *мишку*". Такі правила відповідають вимогам визначення правил контекстно-залежної граматики "$\xi_1 A \xi_2 \rightarrow \xi_3 \alpha \xi_4$". Тут $\xi_1 = \xi_3 = \xi_4 = \varepsilon$ (пустий рядок), $A = N$, $\xi_2 = \sigma$ (або ó), а α — це або слово "*кішка*" або слово "*мишка*".

Повністю контекстно-залежна граматика мови речення визначається як:

$G = (T, N, S, P)$

$T = \{$ *кішка*, *спіймала*, *мишку* $\}$

$N = \{ S, NP, VP, N, V, \sigma, ó \}$

$S = S$

$P = \{ S \rightarrow NP\ \sigma\ VP\ ó$

$NP \rightarrow N$

$VP \rightarrow V\ NP$

$V \rightarrow$ *спіймала*

$N\ \sigma \rightarrow$ *кішка*

$N\ ó \rightarrow$ *мишку* $\}$

Граматика породжує одне речення, правильне з точки зору синтаксису і семантики української мови:

$$S \Rightarrow NP\ \sigma\ VP\ \acute{o} \Rightarrow N\ \sigma\ VP\ \acute{o} \Rightarrow N\ \sigma\ V\ NP\ \acute{o} \Rightarrow N\ \sigma\ V\ N\ \acute{o} \Rightarrow$$

$$\Rightarrow кішка\ V\ N\ \acute{o} \Rightarrow кішка\ V\ мишку \Rightarrow кішка\ спіймала\ мишку\ .$$

Висновки. Приклад 1.6 демонструє що контекстно-залежна граматика може мати менше нетерміналів і (або) правил ніж еквівалентна контекстно-вільна. Так, в неї немає потреби вводити додаткові нетермінали виду *Nns*, *Nao* та інші. Створення граматики спрощується також завдяки узагальненню деяких її правил. Так, на відміну від граматики з прикладу 1.5, правила "$NP \rightarrow N$" і "$VP \rightarrow V\ NP$" залишаються максимально абстрактними (тобто, узагальненими) і незмінними навіть після введення контекстів в граматику.

Контексти дозволяють вирішувати всі основні проблеми моделювання природної мови — проблеми відмінків, дієвідміни, часу, артиклів, дій суб'єктів і об'єктів, характеристик предметів та інші.

Тим самим часом, концептуально контекстно-залежна граматика не стає простішою ніж контекстно-вільна. Контекстів, в яких використовуються слова в природних мовах багато, тому розробка контекстно-залежної граматики мови є складною задачею не зважаючи на потенційно меншу кількість правил. До того ж, комп'ютерні алгоритми розпізнавання речень контекстно-залежних мов є неефективними. Продуктивність таких алгоритмів швидко падає в залежності від кількості контекстів і слів в реченнях. Тому в сучасних програмах комп'ютерної обробки природних мов, контекстно-залежні граматики і мови майже не використовуються.

Наступні два підрозділи розглядаються шляхи вирішення проблеми складності розробки контекстно-вільних і контекстно-залежних формальних граматик. Одним шляхом є спроба виявлення і узагальнення правил породження речень природних мов (підрозділ

1.4). Іншим шляхом є виведення правил граматики з речень мови методом “знизу-наверх” (підрозділ 1.5).

4. ТРАНСФОРМАЦІЙНІ ГРАМАТИКИ

Контекстно-вільні і контекстно-залежні граматики виходять з того положення, що всі правила мови як синтаксичні так і семантичні повинні знайти своє відображення в правилах граматики.

В 1950х роках Хомський висунув ідею, що людина має два рівні механізму мовної компетенції. Один є закладеним в неї при народженні і реалізує “глибинні” синтаксичні структури, які є найбільш узагальненими і не залежними від конкретної мови. Другий механізм виникає в результаті навчання і відповідає за перетворення або трансформацію речень утворених за допомогою глибинних структур в речення конкретної мови. Граматика, що реалізує як глибинні синтаксичні правила так і трансформаційні правила, буде називатися **трансформаційною**.

В граматиках з прикладів 1.5 і 1.6 деякі правила не є притаманними тільки українській мові, а є характерними для багатьох природних мов одразу. Наприклад, це правило, що речення складається зі фрази іменника і фрази дієслова. Також, що фраза дієслова складається зі дієслова і фрази іменника. Такі правила дозволяють породжувати основу (або “скелет”) майбутнього речення. А вже потім ця основа може бути перетворена на правильне речення конкретної мови додаванням, наприклад, правильних закінчень іменників згідно їх роду і числу.

Схожим чином з основи речення можна шляхом трансформації отримати пасивний залог, бо він в багатьох випадках лише змінює порядок слів в реченні додаючи за потребою додаткові слова.

Приклад 1.7. Трансформаційна граматика для речень виду

кішка спіймала мишку

мишка була спіймана кішкою

буде складатися з двох частин — контекстно-залежної граматики для основи речення і трансформаційних правил для перетворення основи на речення активного чи пасивного залогу.

Контекстно-залежна граматика буде мати правила породження основи речення:

$S \rightarrow NP\ Subject\ VP\ Object$

$NP \rightarrow N$

$VP \rightarrow Verb\ NP$

$Verb \rightarrow Aux\ V$

$Aux \rightarrow Modal\ Past \mid Past$

$V \rightarrow спійм$

$N \rightarrow кіш$

$N \rightarrow миш$

Ці правила побудовано таким чином, аби вони давали основу речення, придатну для перетворення в правильне речення як активного так і пасивного залогу. Контексти *Subject* і *Object* введені для породження відмінкових закінчень, контекст *Aux* — для породження модального дієслова (якщо потрібно) і закінчення дієслова. Отже, основа речення, побудованого за цими правилами породжується як:

$$S \Rightarrow NP\ Subject\ VP\ Object \Rightarrow N\ Subject\ Verb\ NP\ Object \Rightarrow$$
$$\Rightarrow N\ Subject\ Aux\ V\ N\ Object\ .$$

Для трансформації основи в речення активного залогу потрібне правило виду:

$N_1\ Subject\ Aux\ V\ N_2\ Object \rightarrow N_1\ ка\ Aux\ V\ N_2\ ку$

$Past\ V \rightarrow V\ Past$

$V\ Past \rightarrow V\ ала$

Тут контексти *Subject* і *Object* замінюються на морфеми, що додають

до іменників потрібні за правилами мови закінчення (і суфікси), а допоміжна частина до дієслова замінюється на його закінчення відповідно до часу дієслова.

Користуючись такими трансформаційними правилами, можна породити правильне речення активного залогу:

N Subject Aux V N Object => N_1 ка Aux V N_2 ку =>

=> кіш ка Past V миш ку => кіш ка V Past миш ку =>

=> кіш ка спійм ала миш ку => кішка спіймала мишку

Для породження речення пасивного залогу потрібна одна трансформація з речення активного залогу і ще одна трансформація для породження слова “була”:

N_1 ка Aux V N_2 ку → N_2 ка Aux V N_1 кою

Modal V Past → була V ана

Породження правильного речення пасивного залогу має вигляд:

N Subject Aux V N Object => N_1 ка Aux V N_2 ку =>

=> N_2 ка Aux V N_1 кою => миш ка Modal Past V кіш кою =>

=> миш ка Modal V Past кіш кою => миш ка була V ана кіш кою =>

=> миш ка була спійм ана кіш кою => мишка була спіймана кішкою .

Висновки. З одного боку, трансформаційні граматики дозволяють дещо спростити контекстно-залежну граматику мови, але доповнення граматики трансформаційними правилами (так як це показано в прикладі 1.7) не обов'язково веде до спрощення граматики.

Однозначно можна стверджувати лише те, що трансформаційні граматики є найбільш потужним інструментом моделювання природних мов, бо трансформаційні правила буквально дають можливості виконати будь-яке перетворення як частини речення так і всього речення.

На сьогодні потужність трансформаційних граматик не використовується в комп'ютерних програмах. Тим не менш, ми

повинні мати їх на увазі, адже їх потенційні можливості і гнучкість моделювання здатні допомогти в рішенні майбутніх задач комп'ютерної обробки природних мов.

5. ГРАМАТИКИ ЗВ'ЯЗКІВ

Як зазначалось у висновках до підрозділу 1.1, формальна граматика, що моделює природну мову, не обов'язково повинна бути побудована “зверху-вниз” від найбільш узагальнених правил конструкції речень до конкретних слів в них.

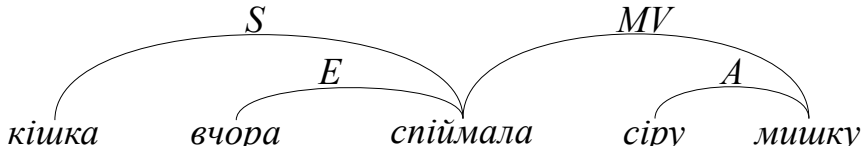
Будь-яка природна мова надає величезний фактичний матеріал, в якому реалізовані всі правила цієї мови, — речення мови. Тобто, на основі речень можна дедуктивним способом вивести правила мови. Граматика зв'язків надає саме таку можливість.

Для більшості природних мов характерно явище проективності — якщо провести лінії між словами, які зв'язані між собою або синтаксично або за смислом, то такі лінії ніколи не перетинаються. **Граматика зв'язків** — це формальна граматика, яка визначає список терміналів (слів) мови і їх можливі зв'язки з іншими словами, при чому послідовність слів є правильним реченням мови, якщо виконуються три умови:

1) зв'язки між словами в реченні не перетинаються (проективність);
2) всі слова в реченні зв'язані один з іншим (зв'язність);
3) виконані всі умови щодо зв'язків кожного слова.

Приклад 1.8. В реченні “*кішка спіймала мишку*” ми маємо два зв'язки між словами. Перший — між іменником-суб'єктом дії і дієсловом, який визначає цю дію. Позначимо цей тип зв'язку літерою *S* (Subject). Другий — між дієсловом і його іменником-об'єктом дії, який позначимо літерою *O* (Object). Ці зв'язки ілюструє граф:

Кожне слово може мати більше ніж один зв'язок, при чому деякі зв'язки можуть бути необов'язковими. Це ілюструє більш складний приклад для речення "*кішка вчора спіймала сіру мишку*":



Тут замість зв'язку *O* між дієсловом-дією і його іменником-об'єктом використаний зв'язок *MV* між дієсловом і його іменником-об'єктом з характеристикою (прикметником). Слово "*спіймала*" має ще один зв'язок *E* наліво. *E* позначає зв'язок між дієсловом і його прислівником-модифікатором перед ним. Зв'язок *A* поєднує іменник і прикметники, які його характеризують.

Два наведених вище приклади зв'язків демонструють, що деякі типи зв'язків для слів в реченні є необхідними: *S*, *O* (або замість нього *MV*). Деякі зв'язки є необов'язковими (*A* і *E),* бо прикметників перед іменником може і не бути, так само як і прислівників перед дієсловом. Такі зв'язки як *A* можуть повторюватись, бо в реченні може бути більш ніж один прикметник перед іменником.

В природній мові зв'язків більше ніж наведено тут в прикладі (107 для англійської). Повний перелік зв'язків можна знайти в інтернеті за адресою http://www.link.cs.cmu.edu/link/dict/summarize-links.html.

Правила запису граматики зв'язків. Для формального запису граматики слід перерахувати слова і вказати всі можливі зв'язки і напрямок зв'язків за наступними правилами:

- напрямок позначається знаком "+" якщо зв'язок іде направо від слова (наприклад, "*кішка*: *S*+");

- напрямок позначається знаком “–” якщо зв'язок іде наліво (наприклад, “*мишку*: *O*−”);
- якщо слово має більш одного зв'язку, то вони об'єднуються знаком “&” (наприклад, “*спіймала*: *S*− & *O*+”);
- якщо слово має більше одного зв'язку в ліву сторону, то зв'язки перелічуються в порядку від найбільш далекого слова до найбільш ближнього (наприклад, “спіймала: *S*− & *E*−”);
- якщо слово має більше одного зв'язку в праву сторону, то зв'язки перелічуються в порядку від найбільш ближнього слова до найбільш далекого;
- якщо зв'язок необхідний, але може бути різних типів, то такі типи об'єднуються оператором “OR” (наприклад, “*мишку*: *MV*− OR *O*−”);
- якщо зв'язок необов'язковий, то він записується в фігурних дужках (наприклад, “*спіймала*: { *E*− }”);
- якщо зв'язок повторюється, то перед ним ставиться знак “@” (наприклад, “*мишку*: *@A*−”).

Приклад 1.8 (продовження): За вказаними вище правилами, повністю граматика зв'язків для мови речень “*кішка спіймала мишку*” і “*кішка вчора спіймала сіру мишку*” записується як:

кішка: *S*+;

вчора: *E*+;

спіймала: *S*− & { *E*− } & (*O*+ OR *MV*+);

мишку: (*O*− OR *MV*−) & { *@A*− };

сіру: *A*+;

Граматику з прикладу 1.8 легко розширити аби вона породжувала більше схожих речень. Так речення “*кішка спіймала сіру спритну*

мишку” буде породжуватись з граматики якщо додати правило виду “*спритну*: *A*+”. А речення “*кішка спіймала пташку*” буде правильним при наявності правила “*пташку*: (*O*− OR *MV*−) & { *@A*− };”.

Важливо відмітити, що в обох наведених випадках граматика розширюється новими словами, а не зв'язками. Зв'язки для слова “*спритну*” будуть тими ж самими, що і для слова “*сіру*”. Аналогічно, зв'язки для слів “*мишку*” і “*пташку*” будуть однакові.

Виявляється, що зв'язки в граматиці є однакові для всіх морфологічно схожих слів, тобто слів з однаковою грамемою. **Грамема** — це рядок, який складається з переліку літер, кожна з яких позначає морфологічну ознаку слова — частину мови, рід, число, відмінок та інші.

Приклад 1.8 (продовження): Слова “*мишку*” і “*пташку*” — це іменники (n, Noun), позначають живу істоту (l, Live), жіночого роду (f, Female), однини (s, Single), в знахідному відмінку (v, accusatiVe). Грамема для обох слів — *nlfsv*.

“*кішка*” — це іменник (n), позначає живу істоту (l), жіночого роду (f), однини (s), в називному відмінку (i, subjectIve case). Грамема слова — *nlfsi*.

“*спіймала*” — це дієслово (v, Verb), доконаного виду (s), перехідне (p), дійсної застави (d), минулого часу (p, Past), жіночого роду (f), однини (s). Грамема слова — *vspdpfs*.

“*сіру*” і “*спритну*” — це прикметники (a, Adjective), жіночого роду (f), однини (s), в знахідному відмінку (v). Грамема для обох слів — *afsv*. Розширена граматика з новими словами і грамемами буде записана як:

кішка.nlfsi: *S*+;
спіймала.vspdpfs : *S*− & { *E*− } & (*O*+ OR *MV*+);
мишку,пташку.nlfsv: (*O*− OR *MV*−) & { *@A*− };

сіру,спритну.afsv: *A*+;

Граматику прикладу 1.8 можна розширювати і надалі, додаючи нові слова з однаковою грамемою в перелік слів для кожного списку зв'язків. Повна граматика зв'язків для природної мови буде складатися з словників слів, згрупованих за морфологічною ознакою (грамемою) і переліком зв'язків для кожного словнику (кожної грамеми).

Отже, граматика зв'язків має дві важливі особливості, які значно спрощують побудування такої граматики для природної мови:

- граматика будується "знизу-наверх", коли на основі речень відтворюються граматичні правила;
- граматику надзвичайно легко розширювати, адже для кожного нового слова треба лише визначити його місце в словниках, не вивчаючи його зв'язки і не додаючи нових зв'язків.

Розроблена граматика зв'язків природної мови буде мати ще одну особливість — еквівалентність контекстно-вільній граматиці. Для кожного речення, для якого є діаграма зв'язків, можна відтворити його контекстно-вільну граматику за простим алгоритмом. Таким чином мова, яку визначає граматика зв'язків, є як найменше, контекстно-вільною.

Висновки. Отже, як інструмент комп'ютерної обробки природних мов граматика зв'язків буде не тільки простою в розробці, але і мати всі переваги контекстно-вільних граматик (поширеність і швидкість комп'ютерних алгоритмів).

Задача розпізнавання речення природної мови буде вирішуватись як задача побудування діаграми зв'язків і перевірка її на відповідність умовам проективності і зв'язності згідно до визначення граматики зв'язків.

Задача генерації речення природної мови буде вирішуватись у той

самий спосіб, що і для контекстно-вільних граматик — породженням речень з кореневого символу еквівалентної контекстно-вільної граматики.

Розробка формальної граматики української мови. На кінець 2008-го року формальної граматики української мови у вільному доступі не існує. Тому, на відміну від англійської мови, з усіх задач комп'ютерної обробки української мови вирішено лише три — пошук в тексті, перевірка правопису і статистичний переклад. Як буде показано в другому розділі, лише для таких задач наявність формальної граматики не обов'язкова. Для подальшого розвитку комп'ютерних технологій обробки української мови конче необхідно створення формальної граматики.

Розробка граматики зв'язків вважається найперспективнішим способом отримати таку формальну граматику. Але для цього необхідні наступні передумови:

- повинен бути зібраний національний корпус української мови, в якому би містились всі відомі правильні речення мови;
- корпус повинен мати морфологічну анотацію, тобто для кожного слова в кожному реченні повинні бути вказані грамеми.

Зібрання національного корпусу проводиться Українським мовно-інформаційним фондом Національної академії наук України (http://ulif.org.ua). За станом на кінець 2008-го року обсяг корпусу становить понад 43 млн. слововживань, але морфологічну анотацію має лише декілька відсотків всіх речень. Також корпус не знаходиться у вільному доступі. Таким чином, розробка граматики зв'язків української мови і досі залишається задачею для майбутніх прикладних лінгвістів.

В світі найбільш повною граматикою зв'язків є граматика англійської мови, розроблена Деніелом Слетором і Девідом

Темперлеєм з університету Карнегі-Мелон на протязі понад 10 років. Граматика випущена під вільною ліцензією і може бути використана при розробці будь-яких комп'ютерних програм обробки англійської мови. Повну інформацію про граматику, програму граматичного розбору і приклади її використання можна знайти на сайті http://www.link.cs.cmu.edu/link/.

Розроблена також граматика російської мови (Сергій Протасов). Граматика досі знаходиться в розробці, хоча і є достатньо повною. Граматика і програми граматичного розбору доступні для придбання. Загальний опис граматики і демонстраційна програма побудування діаграми зв'язків представлені на сайті http://slashzone.ru/parser.

6. СТАТИСТИЧНІ МОДЕЛІ ПРИРОДНИХ МОВ

На початку попереднього підрозділу було зроблено висновок, що речення мови несуть в собі всю інформацію про неї, а аналіз цієї інформації дозволяє відтворити правила мови з її речень. Лінгвістичний спосіб аналізу, тобто відтворення переліку граматичних правил, було розглянуто в розділі 1.5 про граматики зв'язків. Але існує і інший спосіб аналізу — статистичний, тобто знаходження математичних закономірностей всередині речень і фіксація цих закономірностей в формулах. **Набір статистичних закономірностей, що спостерігаються в реченнях, буде складати статистичну модель мови.**

Приклад 1.9. Розглянемо три речення:

S_1: *кішка спіймала мишку*

S_2: *кішка просвердлила мишку*

S_3: *кішка зацахала мишку*

Для людини очевидно, що речення S_2 і S_3 не є правильними реченнями української мови. Людина скаже, що, по-перше, слова "*зацахала*" не існує, а по друге — кішки звичайно не "*свердлять*" мишок. В обох випадках висновок робиться на основі досвіду. Судження "не існує" для слова "*зацахала*" означає, що людина бачила в своєму житті деяку кількість речень, і в жодному з них слова "*зацахала*" не було. Так само і зі словосполученням "*кішка просвердлила*", — у всіх знайомих висловлюваннях зі словом "*просвердлила*", ніде "*кішка*" не виконувала цієї дії.

Формалізуючи дії людини, можна сказати, що вона застосовує статистику, тобто дані про частоту тих чи інших слів і словосполучень, аби зробити своє судження. Говорячи статистично, для людини вірогідність речення S_2 низька в порівнянні з реченням S_1, а вірогідність речення S_3 дорівнює нулю.

Виявляється, що можна побудувати модель мови, яка би робила судження про речення мови подібно до людини з прикладу 1.9 з використанням статистики. Ця "статистична" модель буде призначати кожному реченню вірогідність в такий спосіб, що правильні речення мови будуть мати найвищу вірогідність. Так само, як і людина, статистична модель буде оцінювати вірогідність речень через вірогідність зустріти слова в тому порядку, в якому вони є в реченні.

Треба відмітити, що статистичний принцип суттєво відрізняється від описаних раніше способів. Розпізнавання речень формальними граматиками є перевіркою їх відповідності до граматичних правил. На відміну від такого "лінгвістичного" способу, статистичні моделі судять про правильність речення тільки на основі вірогідності їх існування.

Описана статистична модель є ще одним способом розпізнавати і аналізувати речення мови, тобто вирішувати одну з задач механізму

мовної компетенції, необхідного комп'ютерам для розуміння природних мов.

Визначення. Вірогідність речення $P(S)$ — це вірогідність події, при якій ми послідовно зустрічаємо всі слова речення один за одним.

Позначимо речення S як послідовність слів від w_1 до w_N:

$$S = w_1\ w_2\ w_3\ ...\ w_N\ .$$

Тоді вірогідність речення $P(S)$ — це вірогідність послідовно зустріти в реченні слово w_1, потім w_2, потім w_3 і так далі до w_N:

$$P(S) = P\ (w_1,\ w_2,\ w_3,\ ...\ w_N).$$

З теорії ймовірності відомо, що вірогідність послідовності подій A і B розраховується за формулою:

$$P(A,B) = P(A) \cdot P(B|A),$$

де $P(A)$ — вірогідність події A, $P(B|A)$ — вірогідність події B при умові що подія A відбулася.

Виходячи з цього, вірогідність речення $P(S)$ буде складатися з добутку вірогідності $P(w_1)$ зустріти слово w_1 на початку речення на вірогідність $P(w_2|w_1)$ зустріти слово w_2 за w_1 на вірогідність $P(w_3|w_1,w_2)$ зустріти слово w_3 за словами w_1 *і* w_2 і так далі до $P(w_N|w_1,w_2,...,w_{N-1})$:

$$P(S) = \prod_{n=1}^{N} P(w_n | w_1, w_2, ... w_{n-1})\ \ .$$

Це рівняння отримало назву n-грамної статистичної моделі мови. Модель названа "n-грамною" тому, що в неї вірогідність слова n-го слова визначається вірогідністю всієї послідовності слів до нього, починаючи з першого.

Приклад 1.10. Вірогідність того, що речення S_1 = "*кішка спіймала мишку*" за n-грамною моделлю буде розраховуватися як

добуток трьох вірогідностей:

$$P(S_1) = P(кішка) \cdot P(спіймала \mid кішка) \cdot P(мишку \mid кішка, спіймала).$$

Для розрахунку в числах потрібні дані про вірогідності $P(кішка)$, $P(спіймала \mid кішка)$ і $P(мишку \mid кішка, спіймала)$, які розраховуються за наступними залежностями.

Вірогідність слова "*кішка*" на початку речення:

$$P(кішка) = \frac{кількість\ речень,\ де\ 'кішка'\ стоїть\ на\ початку\ речення}{кількість\ речень,\ в\ яких\ є\ слово\ 'кішка'}.$$

Вірогідність слова "*спіймала*" після слова "*кішка*":

$$P(спіймала \mid кішка) = \frac{кількість\ речень,\ де\ 'спіймала'\ стоїть\ після\ 'кішка'}{кількість\ речень,\ в\ яких\ є\ слово\ 'спіймала'}.$$

Вірогідність слова "*мишку*" після "*кішка спіймала*":

$$P(мишку \mid кішка,\ спіймала) = \frac{кількість\ речень,\ де\ 'мишку'\ стоїть\ після\ 'кішка\ спіймала'}{кількість\ речень,\ в\ яких\ є\ слово\ 'мишку'}$$

Говорячи теоретично, можна легко підрахувати всі необхідні кількості для цих формул з корпусу мови. На практиці задача підрахунку кількостей виявляється складнішою, адже займає багато часу на обчислення через велику кількість слів і речень в корпусі. За різними підрахунками, сучасна природна мова має близько 300 000 слів. Припустимо, що середня довжина речень мови складає 15 слів. Тоді кожне слово може зустрічатися після 14-ти інших слів в реченні, взятих з набору в 300 тисяч слів. Тоді теоретична максимальна кількість вірогідностей, які треба підрахувати для моделі, буде складати близько $300\,000!/(300\,000-15)! \approx 10^{83}$. Звичайно, на практиці кількість слів в корпусах мови значно менша, також слова зустрічаються один за одним не у всіх комбінаціях. Тим не менш, складність задачі залишається "факторіальною" (дивіться докладніше про проблему складності обчислень в додатку А).

Аби спростити розрахунки, вводять допущення, що не всі слова

перед даним словом істотно впливають на його вірогідність. Модель, в якій враховується тільки два попередніх слова, називається триграмною. Якщо враховується тільки одне — то модель є біграмною.

Визначення. Біграмна статистична модель мови — це модель, яка розраховує вірогідність слова в реченні тільки на основі одного попереднього слова, тобто вірогідність усього речення розраховується як:

$$P(S)=\prod_{n=1}^{N} P(w_n|w_{n-1}) \quad .$$

Продовження прикладу 1.10. Біграмна модель для речення S_1 = "*кішка спіймала мишку*" буде складатися з добутку трьох ймовірностей:

$$P(S_1) = P(\textit{кішка}) \cdot P(\textit{спіймала} \mid \textit{кішка}) \cdot P(\textit{мишку} \mid \textit{спіймала}).$$

Приблизний розрахунок на основі даних з пошукової системи Google дає:

$P(\textit{кішка}) = 225\ 000 / 263\ 000 = 0{,}86;$

$P(\textit{спіймала} \mid \textit{кішка}) = 891 / 10\ 300 = 0{,}09;$

$P(\textit{мишку} \mid \textit{спіймала}) = 437 / 64\ 200 = 0{,}007;$

$P(S_1) = 0{,}86 \cdot 0{,}09 \cdot 0{,}007 = 0{,}005.$

Аналогічний розрахунок для речення S_2 = "*кішка просвердлила мишку*" дає:

$P(\textit{кішка}) = 225\ 000 / 263\ 000 = 0{,}86;$

$P(\textit{просвердлила} \mid \textit{кішка}) = 0 / 69 = 0;$

$P(\textit{мишку} \mid \textit{просвердлила}) = 0 / 64\ 200 = 0;$

$P(S_2) = 0{,}86 \cdot 0 \cdot 0 = 0.$

Якщо перше речення (S_1) є вірогідним, то друге (S_2) — зовсім невірогідним з точки зору біграмної моделі української мови. Таким чином, статистична модель на цифрах підтверджує судження людини, що друге речення — не є правильним реченням мови.

Використання статистичних n-грамних моделей. На відміну від лінгвістичних моделей, які потребують до 10 років на розробку, статистичні моделі можна отримати за декілька місяців. Для їх розробки потрібно лише:

1. зібрати корпус мови без будь-якої анотації (ні морфологічний ні синтаксичний аналіз не потрібні для розрахунку вірогідностей);
2. розрахувати за допомогою комп'ютерної програми всі біграми, які зустрічаються в тексті і отримати перелік вірогідностей слів за іншими словами в реченнях.

Ця простота обумовлює надзвичайну популярність і поширеність статистичних методів обробки текстів на природних мовах. Вони є єдиним відомим способом рішення задачі розпізнавання звуків мови і переважно використовуються для витягу інформації з текстів і перекладу. Більшість алгоритмів обробки текстів, описаних в другому розділі цього посібника використовують саме статистичні моделі.

Приклад 1.11. Розглянемо як використовуються статистичні моделі для розпізнавання речень, які вимовляються людиною.

Коли людина говорить, вона зазвичай не робить пауз між словами в реченні. Будь-яка програма розпізнавання вимовної мови повинна транслювати звуки в букви і отримати речення у вигляді одного великого слова. Потім вона повинна розбити це речення на окремі слова. Саме в цей момент стає потрібна статистична модель.

Наприклад, програма слухала людину і розпізнала таке речення:

X = *немаволіїхати*

Існує два способи розбити його на окремі слова:

X_1 = *нема волі їхати*

X_2 = *не мав олії хати*

За біграмною моделлю (приблизні дані з пошукової системи Google) вірогідність речень буде розраховуватись як:

$P(X_1) = P(нема) \cdot P(волі \mid нема) \cdot P(їхати \mid волі) =$

$= 2100000 / 4850000 \cdot 401 / 1710000 \cdot 4 / 776000 = 5{,}2 \cdot 10^{-10}$;

$P(X_2) = P(не) \cdot P(мав \mid не) \cdot P(олії \mid мав) \cdot P(хати \mid олії) =$

$= 7000 / 37100000 \cdot 531000 / 2830000 \cdot 1 / 300\,000 \cdot 0 / 537000 = 0$;

Хоча вірогідність першого речення дуже низька, воно все ж таки є вірогідним на відміну від другого. Тому програма розпізнавання вибере перше речення, знову так, як це би зробила людина.

Висновки. Отже, статистичні моделі є найбільш простим і легким для застосуванням засобом обробки природних мов на комп'ютері. Їх переваги — це відсутність необхідності складного та трудомісткого лінгвістичного аналізу текстів і побудування формальних правил мови, простий спосіб розрахунку параметрів моделей (біграмних і триграмних), швидкість розрахунку на комп'ютерах.

Для статистичних моделей задача розпізнавання речення природної мови є задачею обчислення вірогідності всіх варіантів побудови речення і вибору найбільш вірогідного з них. Задача генерації речення природної мови — це задача випадкового вибору найбільш вірогідної комбінації зі слів.

Дані вірогідностей слів для біграмних і триграмних моделей більшості мов часто вже підраховано. Вони є або у вільному доступі, або їх можна придбати. Для англійської мови компанія Google розповсюджує за ціною носіїв інформації дані п'ятиграмної моделі

(яка також включає в себе біграми і триграми). Дивіться http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html. На жаль, дані для української мови не розповсюджуються, тобто побудування загальнодоступної статистичної моделі української мови залишається задачею майбутніх прикладних лінгвістів.

7. ПОШУК В ТЕКСТАХ

На перший погляд пошук в текстах не є складною задачею. Адже все, що треба зробити — це знайти документи, в яких зустрічаються слова з запиту, і відсортувати ці документи за частотою слів. Перший документ зі списку буде найбільш релевантним, тобто тим, що найбільш відповідає запиту.

Насправді такий простий пошук буде мати дві основні проблеми.

1. Лінгвістична проблема — не будуть знайдені документи, в яких пошукове слово зустрічається в інших словоформах (наприклад, в іншому числі або відмінку). Не будуть знайдені синоніми і гіпоніми (більш конкретні поняття, підвиди). Помилки правопису як в документах так і в пошуковому запиті також будуть впливати на якість пошуку.

   Наприклад, пошук для слова "*машина*" не буде включати документи, що містять "*машини*", "*машину*", "*машиною*" та інші. Також не будуть знайдені документи з синонімом "*автомобіль*" і документи з гіпонімами "*седан*" і "*джип*".

2. Статистична проблема — сортування документів за частотою слів буде робити найбільш релевантними тексти, в яких містяться часто вживані слова.

   Наприклад, пошук для словосполучення "*американська мавпа*" поверне спочатку документи зі словом "*американська*", адже воно вживається частіше ніж слово "*мавпа*" (близько 546 000 документів проти 70 000 за статистикою пошукової системи

Google).

Практика використання пошукових систем демонструє що найчастіше користувачів цікавлять саме рідкі слова. Інші слова в пошуковому запиті часто є лише характеристикою тих рідких (і важливих) об'єктів, які шукають. Так в нашому прикладі "*американська"* є характеристикою і уточненням поняття "*мавпа*", по якому і відбувається пошук.

Рішення лінгвістичної проблеми пошуку полягає в розширенні пошукового запиту всіма словоформами слів зі запиту і всіма їх синонімами і гіпонімами. Для цього пошукова система повинна мати морфологічний словник і словник типу WordNet (словник-мережа, який поєднує зв'язками слова-синоніми, гіпоніми, гіпероніми та інші).

Наприклад, пошуковий запит з одного слова "*машина*" буде трансформований пошуковою системою в вираз "*машина* АБО *машини* АБО *машині* АБО *машину* АБО *машиною* АБО *автомобіль* АБО *автомобілю* АБО *автомобілем* АБО *седан* АБО *джип* АБО ...". Тобто, всі синоніми, гіпоніми і їх форми включаються в пошуковий запит та об'єднуються логічним оператором "АБО". Таким чином будуть знайдені всі документи, в яких є хоча б одне слово з розширеного запиту.

Вплив помилок правопису можна зменшити пошуком схожих слів і автоматичним пропонуванням користувачеві інших варіантів пошуку.

Наприклад, пошук по слову "*машина*" повинен пропонувати додатковий пошук по слову "*машина*". Цей ефект легко побачити в існуючих пошукових системах, в особливості в Google: http://www.google.com.ua/search?q=машна.

Для пошуку схожих слів використовуються фонетичні алгоритми Soundex і Metaphone. Всі фонетичні алгоритми були розроблені спочатку для англійської мови і потім адаптовані для української зі

втратою точності (особливо у випадку з Soundex). Тому розроблені уточнені алгоритми типу D-M Soundex і Double Metaphone.

Всі ці алгоритми розраховують числовий або символьний код для кожного слова. Слова, які звучать подібно один до одного будуть мати однаковий код.

Якщо слово в пошуковому запиті написано з помилкою (тобто воно не знайдено в словнику), то серед слів в словнику шукаються слова з однаковим кодом і пропонуються користувачеві як альтернатива.

Алгоритм 2.1 (Soundex). Для слова розраховується код з однієї літери і трьох цифр:

a) перша буква слова записується в код;
b) кожна приголосна (окрім першої літери) замінюється на число за такими правилами:
   б, в, п, ф => 1
   г, ж, з, к, с, х, ц => 2
   д, т => 3
   л => 4
   м, н => 5
   р => 6
   ч, ш, щ => 7;
c) кожна голосна вилучається;
d) сусідні однакові цифри в коді замінюються на одну;
e) в якості коду береться перша літера і три цифри одразу за нею;
f) якщо цифр менше трьох, дописуються в кінець необхідна кількість нулів.

Приклад 2.1. Наприклад, використаємо алгоритм 2.1 для слова "*машина*":

a) Машина
b) Ма7и5а
c) М75
d) М75
e) М75
f) М750

Для помилкового слова “машна” алгоритм 2.1 поверне той самий код М750, таким чином слово “*машина*” стане альтернативою слову “*машна*” в пошуковому запиті.

Рішення статистичної проблеми пошуку. Алгоритм TF-IDF.

При пошуку кожне слово з пошукового запиту має свою “вагу” — коефіцієнт, який позначає важливість цього слова для результату.

Зазвичай розраховуються два коефіцієнти:

- **TF** (Term Frequency) — частота слова в конкретному документі, яка позначає міру важливості слова в цьому документі;
- **DF** (Document Frequency) — частота слова в усіх документах, яка позначає міру важливості слова в усіх документах.

Вага слова в такому випадку буде дорівнювати множенню TF і DF (записується як TF-DF). Документи, в яких знайдено слова з пошуку сортуються за значеннями ваги кожного зі слів.

Часто вживані слова будуть мати великі значення DF і тому будуть сильно впливати на сортування документів, значно знижуючи точність пошуку.

Карен Джоунс в 1972 році запропонувала найпростіше і найефективніше рішення цієї проблеми — використання інвертованих частот слів в документах IDF=1/DF (Inverse Document Frequency). Інвертована частота навпаки знизить вагу часто вживаних слів, бо вага слова за цим методом буде дорівнювати множенню TF на IDF (записується як TF-IDF).

Алгоритм 2.2 (TF-IDF). Вага слова w_i з пошукового запиту з N слів в документі d_j з корпусу, в якому є D документів, розраховується наступним чином:

a) частота i-го слова в j-му документі:

$$\mathrm{TF}_{\mathrm{ij}} = \frac{n_{ij}}{\sum_{k=1}^{N} n_{kj}},$$

де n_{ij} — число входжень i-го слова в j-й документ,

n_{kj} — число входжень k-го слова в j-й документ,

а знаменник взагалі позначає сукупну кількість входження всіх пошукових слів в j-й документ;

b) інвертована частота слова в усіх документах:

$$\mathrm{IDF}_{\mathrm{i}} = \log\left(\frac{D}{|\{d_j : w_i \in d_j\}|}\right),$$

де $|\{d_j : t_i \in w_j\}|$ — кількість документів, в яких входить i-те слово;

c) вага слова:

$$\mathrm{TF\text{-}IDF}_{\mathrm{ij}} = \mathrm{TF}_{\mathrm{ij}} \cdot \mathrm{IDF}_{\mathrm{j}}$$

Приклад 2.2. Наприклад, проводиться пошук за запитом "*американська мавпа*". Припустимо, що пошукова система має 10000 документів. З них слово "*американська*" знаходиться в 8000 документів, а "*мавпа*" — в 1000.

В позначеннях алгоритму 2.2:

$N = 2$,

$D = 10000$,

$w_1 =$ *американська*,

$w_2 =$ *мавпа*,

$|\{d_j : w_1 \in d_j\}| = 8000$,

$|\{d_j : w_2 \in d_j\}| = 1000$.

Припустимо, що знайдено два документи де зустрічаються оба слова водночас. В одному є одне слово “*мавпа*” і 20 слів “*американська*”. В другому є 2 слова “*мавпа*” і 40 слів “*американська*”.

В позначеннях алгоритму 2.2:

$n_{11} = 20,$

$n_{12} = 10,$

$n_{21} = 1,$

$n_{22} = 2,$

$\sum_{k=1}^{N} n_{kl} = 20 + 1 = 21,$

$\sum_{k=1}^{N} n_{k2} = 10 + 2 = 12.$

За алгоритмом 2.2 ваги слів розраховуються як

$\mathrm{TF}_{11} = \frac{20}{21} = 0{,}95$, $\mathrm{TF}_{12} = \frac{10}{12} = 0{,}83$,

$\mathrm{TF}_{21} = \frac{1}{21} = 0{,}05$, $\mathrm{TF}_{21} = \frac{2}{12} = 0{,}17$,

$\mathrm{IDF}_1 = \log\left(\frac{10000}{8000}\right) = 0{,}09$, $\mathrm{IDF}_2 = \log\left(\frac{10000}{1000}\right) = 1$,

$\text{TF-IDF}_{11} = 0{,}95 \cdot 0{,}09 = 0{,}085$, $\text{TF-IDF}_{12} = 0{,}83 \cdot 0{,}09 = 0{,}075$,

$\text{TF-IDF}_{21} = 0{,}05 \cdot 1 = 0{,}05$, $\text{TF-IDF}_{22} = 0{,}17 \cdot 1 = 0{,}17$.

Для сортування сумарна вага першого документу буде 0,085 + 0,05 = 0,135, а другого — 0,075 + 0,17 = 0,245, тобто другий документ буде вважатися найбільш релевантним при пошуку.

Незважаючи на значно меншу кількість слів “*американська*” в другому документі, наявність просто ще одного слова “*мавпа*” надало йому завдяки інвертованій частоті IDF майже в два рази більше ваги ніж першому документу. Це наочно ілюструє, що використання TF-IDF збільшує цінність рідко вживаного слова “*мавпа*” і виділяє його як головне поняття, яке шукає користувач в тексті.

Приклади використання. Пошукові системи Google (http://google.com), Yandex (http://yandex.ru), Live (http://live.com), Ask (http://ask.com) та багато інших, — всі вони використовують наведені вище лінгвістичні і статистичні алгоритми пошуку в тексті.

Пошук різних словоформ і синонімів, коригування помилок правопису, пріоритет рідко вживаних слів легко продемонструвати, якщо запустити пошук по запитам "*машина*", "*машна*" і "*американська мавпа*" в будь-якій пошуковій системі.

На сьогодні задача пошуку в текстах є вирішеною найбільш успішно. Більше того, на базі пошуку в текстах з мережі Internet було організовано надзвичайно прибутковий бізнес.

Пошукова система Google на сьогодні має дані про більше ніж 1 трильйон інтернет сторінок (точну цифру настільки складно підрахувати, що сам Google не здатен надати таку інформацію). Кожного дня людьми виконується близько 3х мільярдів пошукових запитів в інтернет, з них 2 мільярди — в Google. Компанія використовує надзвичайну популярність пошукового сервісу для розміщення реклами. Коли користувач виконує пошуковий запит, разом з результатами пошуку йому повертається і реклама про товари та послуги, пов'язані зі запитом. Завдяки цьому Google в 2007-му році мав дохід 16 млрд. доларів і прибуток близько 5 млрд. доларів.

Російська компанія Yandex повідомила про прибутки понад 64 млн. доларів в 2007-му році. Українська пошукова компанія Meta (http://meta.ua), яка є третім за поширеністю пошуковим сервісом в Україні, мала близько 100 тис. доларів прибутків за 2007-й рік.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. The Oxford Handbook of Computational Linguistics (Oxford Handbooks in Linguistics) / R. Mitkov, - Oxford University Press, USA, 806p.
2. John Hutchins: Retrospect and prospect in computer-based translation. Proceedings of MT Summit VII, 1999, pp. 30–44.
3. Arnold B. Barach: Translating Machine 1975: And the Changes To Come.
4. The Association for Computational Linguistics What is Computational Linguistics? Published online, Feb, 2005.
5. John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman. Introduction to Automata Theory, Languages, and Computation (2nd Edition), Addison Wesley; 2 edition (November 24, 2000), 521 p.
6. Бук С. Основи статистичної лінгвістики: Навчально-методичний посібник / Відп. ред. проф. Ф. С. Бацевич.— Видавничий центр ЛНУ імені Івана Франка, 2008.— 124 с.
7. Chomsky, Noam, Syntactic Structures. The Hague: Mouton
8. Chomsky, Noam, Language and Mind.
9. Chomsky, Noam, Aspects of the Theory of Syntax. Cambridge: The MIT Press.