УДК 004.422.833(045)

Сич М.Ю.

МОДУЛЬ ПРИЙОМУ КОМАНД НА ПРИРОДНІЙ МОВІ

Національний авіаційний університет

Приведена загальна структура та принципи роботи модуля прийому команд на природній мові – одного з найновіших видів взаємодії людини та комп'ютерних систем

Вступ

Сучасні комп'ютерні технології зробили величезний крок у розробці систем обробки природної мови, на основі яких будуються найрізноманітніші механізми взаємодії людини та машини, розуміння інформації. На сьогоднішній день вони використовуються при побудові експертних систем, автоматизованих інформаційних систем, різноманітних робото-технічних блоків, не кажучи вже про інтелектуальну обробку інформації, системи машинного перекладу та інтерфейси прийому команд.

Проте, незважаючи на значне поширення систем обробки природної мови, вони не мають достатнього рівня розвиненості. Алгоритми обробки у них не є достатньо потужними та надійними. Точність обробки інформації залишає бажати кращого, адже більшість систем не можуть запропонувати достатньо добрих механізмів, щоб впоратись з неточностями інформації, двозначностями та іншим.

Наслідком цього є ускладнення структури таких систем, їх незрозумілість, часто навіть для інженерів, що їх будують та використовують. Отже, ми можемо констатувати, що необхідно розробляти нові алгоритми обробки мови, смислової обробки. Ці алгоритми мають давати високу точність розуміння інформації, давати можливість маніпулювати нею. Тільки на основі таких потужних алгоритмів можна побудувати прості та надійні системи.

Постановка задачі

Необхідно розробити та запропонувати загальну можливу структуру модуля прийому команд на природній мові. Описати основні концепції його роботи, загальний алгоритм функціонування та розглянути можливості реального використання такого модуля.

Модуль прийому команд на природній мові(МПК) призначений для сприймання команд від користувача, їх обробки, переклад у формат, що може бути оброблений певною комп'ютерною системою, а також запуск команди на виконання. Сам модуль представляє собою завершену програму роботи з користувачем, тобто його функціональність розроблена орієнтовно на роботу з людиною.

Функції модуля прийому команд

Даний модуль розроблений для інтеграції у операційні системи сімейства *Microsoft Windows, Unix* та *Linux* і містить набір визначених запрограмованих команд для виконання.

До основних функцій МПК входять безпосередній прийом команд на природній мові від користувача та повна обробка команди, що включає в себе виділення команди, її декодування у формат, що може сприйматись комп'ютером, а також запуск відповідної команди на виконання.

Точний опис операцій, що виконує модуль прийому команд наведений нижче:

1) прийом команди від користувача. Для цих цілей був розроблений простий графічний інтерфейс користувача. Він має елементи, необхідні для прийому команди у форматі текстового рядку, що буде передаватись для обробки;

2) семантична обробка введенного користувачем текстового рядку. Основною метою цієї функції модуля є знаходження основних смислів слів та характеру взаємозв'язків між ними. Результатом даної обробки має стати структура,

що буде містити слова, їх точні змісти та описувати види зв'язків між словами, їх смислове навантаження;

3) виділення команди та її атрибутів. Ця операція представляє собою обробку структури, що була отримана на ви-ході попередньої функції. Задача полягає у пошуку слова, що містить назву команди, її суть. Також необхідно виділити атрибути, обов'язкові для виконання даної команди. Далі відбувається запуск відповідної команди;

4) виконання команди. Задача полягає у безпосередній реалізації операції, що була запитана користувачем.

Модуль побудований як композиція взаємопов'язаних частин, що у подальшому ми будемо називати *блоками*. Кожен з цих блоків має певний набір функцій та використовується для певних цілей.

Список блоків модуля наступний:

1) інтерфейс користувача – ІК;

2) блок перекладу – БП;

3) блок семантичного аналізу – БСА;

4) блок виконання – БВ.

Графічний інтерфейс користувача

Призначений для сприймання команди від користувача та передачі його для обробки у блок перекладу. Оскільки модуль налаштований на роботу з текстовим представленням команди, то інтерфейс користувача може бути простим рядком для вводу інструкції користувачем.

Блок семантичного аналізу

Займається обробкою текстової інформації з метою знаходження точних смислів слів, а також встановлення точних семантичних зв'язків між ними. Блок має справу з природною людською мовою. Він отримує на вхід конструкцію слів, що вводиться користувачем і намагається знайти у цій конструкції зв'язки між словами. Інакше кажучи, цей блок займається знаходженням логічних та семантичних зв'язків між словами у введеній команді.

Для реалізації функцій блоку семантичного аналізу був розроблений алгоритм семантичної обробки словарних конструкцій з метою визначення зв'язків між словами. Цей алгоритм заснований на кодуванні слів, що передбачає відмову від роботи зі словами як з *ASCII* – послідовністю символів та передбачає кодування смислів слів. Слова кодуються у вигляді 64-розрядних чисел, що дає можливість закодувати величезну кількість смислів, а також закласти у код додаткову інформацію, що залежить від частини мови слова.

Це можуть бути вказівки на числа, відмінки, роди. Включивши у код слова також дані про синонімічні ряди отримуємо можливість обробляти команди, що можуть бути введені користувачем з використанням різних словарних конструкцій. До коду слова додаються групи ключових додатків, що вказують на можливі зв'язки слова з іншими. Таких груп додатків у кожного слова дві – одна група вказує на взаємодії з іншими словами, де поточне можу виступати у якості підлеглого, інша група – у якості головного слова. Знаходячи перетини груп ключових додатків слів ми можемо встановити зв'язки між словами, що дає нам можливість обробити зміст команди та знайти її необхідні атрибути.

Отже ми маємо алгоритм, що базується на кодування слів. Для зберігання цих слів була обрана реляційна база даних і закладений словник з відповідними даними про слова.

Блок перекладу

Є керуючим по відношенню до інших частин систем. До його функцій входить запуск інших блоків(графічного ін-терфейсу користувача, блоку семантичного аналізу та блоку виконання), передача необхідних даних цим блокам, викликвідповідних методів обробки і т.д. Алгоритм роботи блоку перекладу можна представити графічно(рис. 1).

Проте основною частиною роботиблоку перекладу є безумовно перетворення обробленої БСА словарної конструкції у безпосередній виклик потрібної функції для виконання інструкцій.

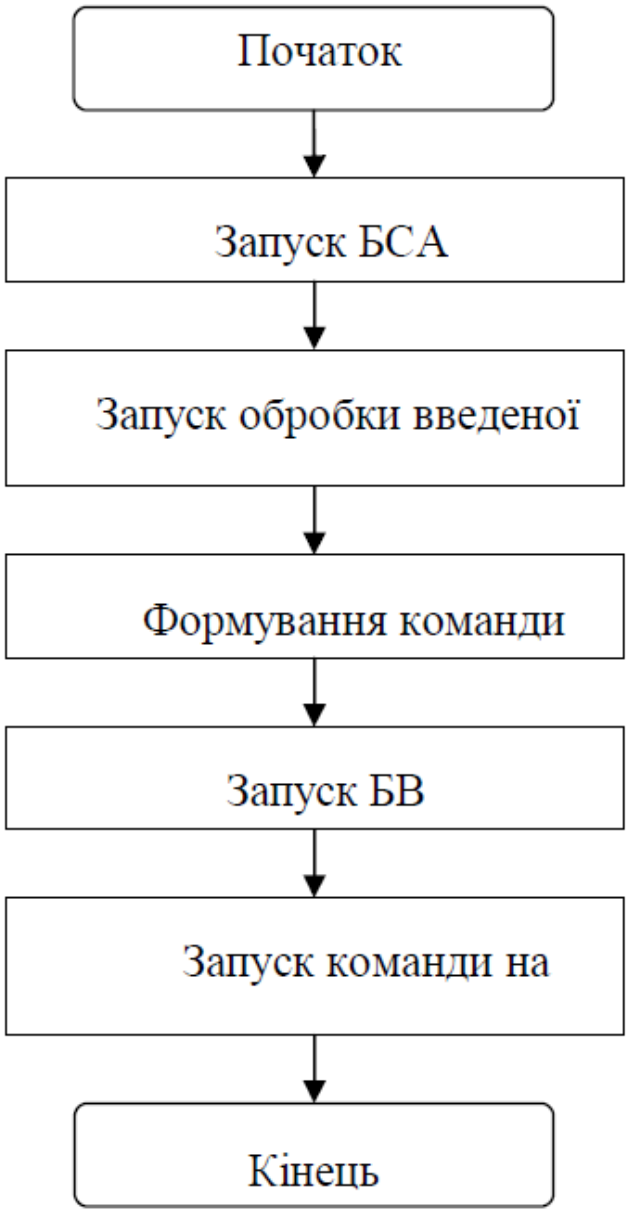

Рис.1. Алгоритм роботи блоку перекладу

Після обробки команди у блоці семантичного аналізу ми маємо змісти всіх введених слів, а також встановлені взаємозв'язки між ними. На основі цих даних блок перекладу встановлює зміст команди та її основні атрибути. Першим кроком є встановлення головного слова у словарній конструкції. Для цього необхідно знайти слова, що не є підлеглим по відношенню до інших слів. Після того, як знайдено таке слово, через словник доступних команд визначається відповідна функція, що відповідає за виконання команди, що позначає головне слово, а також список необхідних атрибутів, що є обов'язковими для коректного виконання команди у комп'ютерній системі.

Кожен атрибут зі списку пов'язаний з характером взаємозв'язку, що міститься між головним словом та словом, що фактично називає значення атрибуту у словарній конструкції. Наведемо приклад.

Нехай введена команда має вигляд:

«Перемістити файл з директорії А у директорію Б». Після обробки команди у БСА встановляться зв'язки типу:

- перемістити (дія-суб'єкт дії) файл;
- перемістити (дія-ознака дії) з директорії;
- перемістити (дія-ознака дії) у директорію;
- директорія (назва) А;
- директорія (назва) Б.

Отримали граф команди, який є орієнтовним, оскільки зв'язки між словами є одно направленими і йдуть від головного слова до залежного.

Головним смисловим словом у цій конструкції є звичайно «перемістити». Це слово означає суть дії, не є підлеглим до жодного іншого і в той же час виступає головним словом для інших. Воно є фактично коренем дерева.

Тепер приведемо структуру типового запису команди зі словнику.

<слово, що означає команду>,
<назва функції реалізації команди>,
<обов'язковий атрибут 1>
<обов'язковий атрибут 2>
. . . . . .
<обов'язковий атрибут N>

Для нашого попереднього прикладу запис у словнику команд буде мати вигляд:

перемістити
Replace
(дія-суб'єкт дії) файл
(дія-ознака дії) з директорії (назва) А
(дія-ознака дії) у директорію (назва) Б,

де *Replace* – назва функції, що виконує команду.

Для більшої зручності та гнучкості словників можна позначити самі слова певними числовими кодами. Теж сааме можна зробити і для характерів взаємозв'язків між словами.

Самі ж зв'язки можна групувати у певні більш складні структури, що будуть

вказувати на місце дії, напрям, суб'єкти, засоби дії і т.д.

Блок виконання

Займається безпосередньо роботою з системою, у яку інтегровано модуль при йому команд. Блок виконання містить у собі набір або бібліотеку команд, що можуть бути запущені у системі. Кожна команда виділена у окрему функцію, що дає можливість у подальшому переписувати модуль та розширювати набір доступних для виконання команд.

Таким чином, запрограмувавши одного разу блок перекладу і заклавши в нього основні базові механізми встановлення змісту команд та знаходження їх атрибутів, ми отримуємо можливість додавати у подальшому нові команди, тобто розширювати список доступних для виконання операцій модуля. Для цього необхідно:

1) додати нові функції у блок виконання;

2) додати відповідний запис про команди у словник.

Після цього блок перекладу зможе динамічно формувати команди на основі вже доступних механізмів без необхідності додавання коду у нього. Фактично дописування потребуватиме лише два фай-ли: список команд і бібліотека функцій.

Реалізація модуля прийому команд

Тепер приведемо деякі рекомендації щодо можливих варіантів реалізації блоків модуля прийому команд на природній мові.

Практична реалізація модуля прийому команд може і бажано відбуватиметься на базі певної об'єктноорієнтовної мови програмування. Це може бути скажімо *C#, JAVA, C++*.

Для використання нового кодування слів у блоці семантичного аналізу їх потрібно записати у словник. Для цього біло побудовано відношення, де закладено відповідності між словами у символьному представленні та кодом 64-розрядного числа. Для збереження відношення використано реляційну СУБД *MySQL Server*. Вона має достатньо потужні механізми роботи з даними, що будуть для нас достатніми. Додатковим чинником для такого вибору служить те, що ця СУБД має урізаний список команд та функціональності, тому використовує менше ресурсів комп'ютера порівняно з *Microsoft SQL Server* чи *Oracle*.

Визначимо структуру таблиці слів. Для початку визначимо дані та поля, що необхідно зберігати.

- слово у текстовому представленні. Це поле необхідне для ідентифікації слова у базі під час конвертування до нового формату. Назвемо поле *word_string* та оберемо тип даних *varchar* (255), що може вмістити слова великої довжини. А оскільки тип даних *varchar* буде зберігати слово точно по кількості символів у ньому, то це дасть нам виграш також у об'ємах використаної пам'яті;

- слово у форматі 64-розрядного числа. Це число, що представляє первинне кодування слова. Для наочності будемо зберігати слово у шістнадцятковому форматі представлення, тому саме число збережемо у вигляді звичайного символьного рядка. Назвемо поле *word_number* іоберемо тип даних *char*(16). Необхідно саме 16 символів для збереження числа, оскільки 64 біти двійкового представлення містить точно - 16 шістнадцяткових розряди;

- кількість додаткових часток у групі вхідних. Назвемо поле *inq_amount* і оберемо тип даних *tinyint*. Тип tinyint має 8 розрядів, отже можу зберігати значення від -127 до 128. Виставимо для поля атрибут *unsigned*, тобто беззнаковий, і таким чином діапазон доступних значень буде (0, 255), що буде цілком достатньо для кодування кількості часток у групі;

- кількість додаткових часток слова у групі вихідних. Назвемо полу *outq_amount*. Тип даних *tinyint* з атрибутом *unsigned;*

- список часток слова групи вхідних. Кількість часток групи для різних слів буде коливатись і не є постійним числом. Очевидним і першим рішенням для збереження є задати кількість полів у

таблиці по кількості питань у групі слова, що має найбільшу їх кількість. Проте таке рішення є дуже невигідним з точки зору оптимальності, а також можлива ситуація коли дозволена кількість полів таблиці – 255, буде недостатньо, щоб закодувати всі питання. Тому для вирішення цієї проблеми ми використаємо великі бінарні поля(*BLOB – binary large object*). Вони є простим набором бітів, у які ми зможемо закодувати будь-яку кількість часток у вигляді потоку байтів. Назвемо поле inqs та оберемо тип даних *tinyblob*, що може вмістити до 255 байтів. Оскільки частки кодуються 64-розрядними числами, то у полі можна зберегти до 31 питання. У разі необхідності можна швидко модифікувати таблицю і розширити поля без ушкодження даних, що там зберігаються;

- список часток слова із групи вихідних. Назва поля – *outqs* і тип даних *tinyblob*.

Поглянемо на поля *inqs* та *outqs* таблиці. В ці поля потоком будуть записуватися частки у вигляді 64-розрядних чисел. Якщо ми створимо додаткову таблицю, де запишемо всі дані по питанням, то зможемо призначити кожній частці унікальний ідентифікатор і записувати у відпові-дні поля не значення самих часток, а ці ідентифікатори. Кількість часток менше 255, тому для ідентифікатора достатньо числа в 1 байт. Отже, кількість часток, що ми зможемо записати у поля *inqs* та *outqs* таблиці *words* виросте у 8 разів.

Структура для таблиці додаткових часток та її поля:

- унікальний ідентифікатор. Назвемо поле *question_number*. Як вже було зазначено вище для кодування кількості часток необхідно 1 байт, тобто тип даних для даного поля *unsigned tinyint*;

- поле символьного представлення додатку. Назва поля – *question_string*, тип даних *char*(40);

- поле 64-розрядного коду частки. Назва поля – *question_digit*. Представимо сам код частки у вигляді шістнадцяткового числа текстовим рядком, отже запишемо у звичайний символьний рядок, тобто - тип даних *char*(16). Під час використання значення рядок просто конвертується у число.

Висновки

В епоху всепоглинаючого поширення комп'ютерної техніки та програмних систем на перше місце виходять задачі створення нових, більш досконалих засобів та механізмів взаємодії людини та машини.

Приведена у статті структура модуля прийому команд простою та зрозумілою. Основною її перевагою є розширюваність та масштабованість, тобто набір доступних для виконання через модуль команд може бути розширений, не змінюючи та не переробляючи основні частини модуля. Це дає змогу використовувати модуль у різних комп'ютерних системах, оскільки її авторам для коректної роботи модуля необхідно буде лише створити відповідну бібліотеку функцій команд.

Список літератури

1. *Вагин В.Н.* Дедукция и обобщение в системе принятия решений. – М.: Наука, 1988. – 384 с.

2. *Гаврилова Т.А.* Логико-лингвистическое управление как введение в управление знаниями // Новости искусственного интеллекта. – 2002. – №6. – С. 36–40.

3. *Леонтьева Н.Н.* Автоматизированный перевод как понимание и реферирование // Прикладные и экспериментальные лингвистические процессоры – Новосибирск, 1981. – С. 21–35.