3. На основі тестування бездротової мережі на злам та на заміну трафіку, який передається ESP 32, сформульовано вектор атаки, який дозволяє клонування пристрою на основі використання компонентів бездротових мереж з відомими вразливостями.

Література

1. Pathan A. K. Securing Cyber-Physical Systems. London: CRC Press, 2015. 236 p.
2. Misra S., Maheswaran M., Hashmi S. Security Challenges and Approaches in Internet of Things. 2017. 106 p.
3. Aziz B., Arenas A., Crispo B. Engineering Secure Internet of Things Systems. Croydon : CPI Group, 2016. 56 p.
4. Gilchrist A. INDUSTRY 4.0 THE INDUSTRIAL INTERNET OF THINGS. Nonthaburi : Apress, 2016. 100 p.
5. Hu F. Security and Privacy in Internet of Things (IoTs). London : CRC Press, 2016. 203 p.
6. Macaulay T. RIoT Control. London: Elsevier, 2017. 13 p.
7. Russell B., Van Duren D. Practical Internet of Things Security. BIRMINGHAM : Packt, 2016. 94 p.
8. The Internet of things with ESP32. URL : http://esp32.net/ (Last accessed: 05.01.2019).
9. Wi-Fi, від англійського Wireless Fidelity, 2009. URL : https://wiki.cuspu.edu.ua/index.php/Wireless_Fidelity/ (Last accessed: 05.01.2019)
10. Колыбельников А. И. Обзор технологий беспроводных сетей. *Московский физико-технический институт*, 2012. 3 с.
11. Макаренко А. Ю., Парфенова А. О., Могильний С. Б. Бездротові технології передачі даних WI-FI, BLUETOOTH TA ZIGBEE. *Національний технічний університет України «КПІ»*, 2010. 127 с.
12. Сайко В. Г., Оксіюк О. Г., Дікарєв О. В. Основи цифрового оброблення сигналів в системах цифрового радіозв'язку. Київ, 2016. 113 с.
13. Шовкута В. А., Флоров С. В. Аналіз механізмів захисту та вразливостей бездротових WI-FI мереж. *ДВНЗ «Національний гірничий університет»*, 2016. 10 с.
14. Parrot Security OS – альтернатива Kali Linux, 2017. URL : https://habr.com/company/pentestit/blog/337712 (дата звернення: 05.01.2019).
15. Airmon-ng, 2014. URL : https://tools.kali.org/wireless-attacks/airmon-ng (дата звернення: 05.01.2019).
16. Airodump-ng, 2017. URL : https://tools.kali.org/wireless-attacks/airodump-ng (дата звернення: 05.01.2019).
17. Besside-ng, 2017. URL : https://tools.kali.org/wireless-attacks/besside-ng (дата звернення: 05.01.2019).
18. Aircrack-ng, 2014. URL : https://tools.kali.org/wireless-attacks/aircrack-ng/ (дата звернення: 05.01.2019).
19. Wireshark, 2014. URL : https://tools.kali.org/information-gathering/wireshark (дата звернення: 05.01.2019).
20. Aireplay-ng, 2017. URL : https://tools.kali.org/wireless-attacks/aireplay-ng (дата звернення: 05.01.2019).
21. Arduino, 2014. URL : https://tools.kali.org/hardware-hacking/arduino (дата звернення: 05.01.2019).

УДК 81.33+004

СИСТЕМА АНАЛІЗУ «USER STORY» НА БАЗІ БІБЛІОТЕКИ NLTK

О. І. Барибін, О. В. Соловей

У сучасному світі прискорення темпів розвитку інформаційних технологій призводить до необхідності використовувати гнучкі підходи до розробки програмних продуктів. Від того, які технології використовує компанія, будуть залежати її конкурентні переваги на ринку. Однією з найбільш актуальних проблем є саме формулювання вимог до програмного

забезпечення. В рамках методології управління проектами для гнучкої розробки Scrum впроваджено форму опису вимог – User Story, тобто формулювання чітких і якісних побажань до майбутнього ПЗ. Тому не тільки невеликі команди розробників, але і великі компанії все більше уваги приділяють розробці користувацьких історій – User Story [1].

Гнучкі методології впевнено намагаються проникнути в великий бізнес, який розуміє необхідність змінювати підхід до організації роботи, щоб вижити в конкуренції з невеликими і гнучкими компаніями. У поточний момент гнучкі методології проникають як в окремі галузі економіки: банківську сферу, телекомунікації, медицину, державний сектор, так і в окремі сфери діяльності: бухгалтерію, маркетинг тощо [2]. Саме тому проблема визначення User Story до розробки програмних продуктів є як ніколи актуальною.

Метою роботи є автоматизація перевірки якості написання User Story на базі бібліотеки NLTK. Для досягнення цілі, передбачено виконати наступні завдання:

- знайти і апробувати бібліотеки Python, що працюють в сфері обробки текстів;
- на основі аналізу шаблону User Story виділити вставні частини та провести токенізацію (лексичний аналіз), перевірку на узгодження й пунктуацію.

Для реалізації програмного забезпечення обрано високорівневу мову програмування загального призначення, яка орієнтована на підвищення продуктивності розробника і читання коду – Python. Синтаксис ядра Python мінімалістичний [3]. У той же час стандартна бібліотека включає великий обсяг корисних функцій. Основними архітектурними рисами є: динамічна типізація, автоматичне керування пам’яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточних обчислень і зручні високорівневі структури даних.

Серед бібліотек, які можна використовувати для обробки природних мов на мові Python можна виділити наступні: spaCy, Stanford CoreNLP, Scikit□learn, Gensim, word2vec та NLTK[4]. Кожна з вище перерахованих бібліотек має свої переваги та недоліки. З урахуванням поставленої цілі, для реалізації програмного забезпечення та особливостей Python було обрано бібліотеку NLTK (Natural Language Toolkit). Важливим є те, що ми можемо визначити місце розташування слова в тексті: скільки слів з самого початку воно з'являється. Ця інформація про місцезнаходження може відображатися з використанням графіка дисперсії. Кожна смуга являє собою екземпляр слова, а кожен рядок являє весь текст [5]. NLTK надає прості у використанні інтерфейси для більш ніж 50 корпоративних і лексичних ресурсів, таких як WordNet, поряд з набором бібліотек обробки тексту для класифікації, токенізації, обробки по мітках, розмітки, синтаксичного аналізу та семантичного мислення, оболонок для промислових бібліотек NLP, і активний дискусійний форум. NLTK був названий «чудовим інструментом для навчання і роботи в області комп'ютерної лінгвістики з використанням Python» і «дивовижною бібліотекою для гри на природній мові».

Для полегшення роботи користувача-замовника, а саме формулювання побажань до розроблюваної системи на основі стандартного формату історії реалізовано шаблонну форму вводу User Story. Тобто, як «користувач», я хочу «дія», для того щоб «мета», де «користувач» – узагальнена роль того, хто виконує дію або отримує від нього користь; «дія» – активність, яка виконується користувачем при взаємодії з системою; «мета» – мета користувача, яка представляє для нього певну цінність, дозволяє отримати вигоду. Тому основна увага приділяється визначенню істинної мети користувача. Іноді виявлена в процесі обговорення мета користувача може стати причиною зміни в історії дії, що була заявленою спочатку

Наступним етапом роботи є виділення вставних частин, для проведення лексичного аналізу (токенізації) та перевірки на узгодження та пунктуацію. Відповідно до стандартного підходу синтаксично аналізу, токенізація – процес аналітичного розбору вхідної послідовності символів на розпізнані групи – лексеми, з метою отримання на виході ідентифікованих послідовностей, званих «токенами» (подібно до угруповання букв в словах. У нашому програмному забезпеченні реалізовані складні токенізатори, які додатково класифікують лексеми по різним типам («ідентифікатор, оператор», «частина мови» і т. д.). Лексичний аналіз використовується в компіляторі та інтерпретаторі вихідного коду. Як правило, лексичний аналіз проводиться з точки зору певної формальної мови або

набору мов. Мова, а точніше її граматика, задає певний набір лексем, які можуть зустрітися на вході процесу.

Таким чином, в результаті роботи реалізовано програмний код, який дозволяє користувачу визначити правильність формулювання User Story, відповідно до розроблюваної системи на основі стандартного формату історії. Тобто в процесі побудови, проводиться аналіз виділених частин, на основі якого й будується User Story. У випадку, якщо користувацька історія сформована некоректно, користувачу пропонуються правильні можливі варіанти, для подальшої роботи.

Література

1. Cohn M. User Stories Applied for Agile Software Development. Boston : Addison-Wesley, 2004. 268 p.

2. Taweh Beysolow II. Applied Natural Language Processing with Python. USA : San Francisco, California, 2018. 150 p.

3. Bird S., Klein E., Loper E. Natural Language Processing with Python. USA : California, 2009. 463 p.

4. Packt Publishing Ltd. Natural Language Processing: Python and NLTK. Birmingham, Mumbai, 2016. 687 p.

5. Lawrence R. Patterns for Splitting User Stories. URL: http://agileforall. com/patterns-for-splitting-user-stories/.

УДК 004.891.3

ОСОБЛИВОСТІ РОЗРОБКИ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ДІАГНОСТИКИ ЗАХВОРЮВАНЬ

М. О. Єпік

Інтелектуальна система (ІС) є взаємопов’язаною сукупністю засобів та методів, яка має можливість зберігання, обробки і видачі інформації, самостійного налаштування своїх параметрів в залежності від стану зовнішнього середовища (початкових даних) і специфіки завдання, що розв’язується. ІС повинна уміти розпізнавати істотні факти з набору фактів; робити висновки на основі наявних фактів і знань; володіти рефлексією, тобто засобами для оцінки результатів власної роботи; пояснювати отриманий результат; узагальнювати, визначаючи схожість між наявними фактами. Інтелектуальні системи успішно вирішують складні завдання у галузі медичної діагностики захворювань. Їх ефективність, а також достовірність отриманих результатів залежать від знань, якими вони володіють.

Інтелектуальна система діагностики захворювань (ІСДЗ) – це інтелектуальна система, що містить знання спеціалістів в царині первинної діагностики захворювань, які можуть приймати експертні рішення в межах своєї компетентності. Особливості розробки (ІСДЗ) пов’язані з функціями, які повинна виконувати система: моделювання мислення лікаря-терапевта при розв’язанні конкретного завдання; використання знань, які має система, для формулювання висновків; володіння здатністю пояснювати отриманий результат (діагноз).

Інтелектуальна система діагностики захворювань (ІСДЗ) складається з наступних модулів: інтерфейсу користувача, бази даних, бази знань, механізму виведення і має структуру, приведену на рис. 1.