ISSN 2663-3051
(ISSN 0555-2656 до 2019 г.)

БИОНИКА ИНТЕЛЛЕКТА

ИНФОРМАЦИЯ, ЯЗЫК, ИНТЕЛЛЕКТ

№ 2 (97) **2021**

НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Основан в октябре 1967 г.

Учредитель и издатель
Харьковский национальный университет радиоэлектроники

Периодичность издания – *2 раза в год*

Харьков • ХНУРЭ • 2021

Научно-технический журнал

«БИОНИКА ИНТЕЛЛЕКТА»

ISSN 2663-3051

Основан Харьковским национальным университетом
радиоэлектроники в 1967 году

Реферирование и индексирование:

Google Scholar

BASE

INDEX COPERNICUS
INTERNATIONAL

Національна бібліотека України
імені В. І. Вернадського

Журнал включен в список научных специализированных изданий Украины
по техническим и физико-математическим наукам
согласно приказа Министерства образования и науки Украины № 820 от 11.07.2016

УДК 004.75 DOi 10.30837/ bi.2021.2(97).01

М.П. Дудник[1], С.Г. Удовенко[2], Л.Е. Чала [3], М.М. Соколовська [4]

[1] студент групи ДСм-21-1,
Харківський національний університет радіоелектроніки,
mykola.dudnyk@nure.ua, ORCID iD: 0000-0002-8259-0221,

[2] доктор технічних наук, професор, завідувач кафедри інформатики та комп'ютерної техніки,
Харківський національний економічний університет ім. С. Кузнеця,
serhiy.udovenko@hneu.net, ORCID iD: 0000-0001-5945-8647

[3] кандидат технічних наук, доцент, доцент кафедри штучного інтелекту,
Харківський національний університет радіоелектроніки,
larysa.chala@nure.ua, ORCID iD: 0000-0002-9890-4790

[4] студентка групи ІТШІ-19-3,
Харківський національний університет радіоелектроніки,
mariia.sokolovska@nure.ua, ORCID iD: 0000-0002-9789-6928

НЕЙРОМЕРЕЖЕВА ТЕХНОЛОГІЯ БАГАТОМОВНОЇ КЛАСИФІКАЦІЇ ЕЛЕКТРОННИХ ТЕКСТІВ

Статтю присвячено розробці технології побудови багатомовних класифікаторів, яка основана на нейромережевій обробці векторного подання текстів, згенерованого за допомогою моделі XLM-RoBerta. Розглянуто переваги використання для векторизації текстів рекуррентної нейронної мережі на основі трансформеру моделі XLM-RoBerta. Наведено схему взаємодії розробленого класифікатору на основі мережі LSTM з моделлю векторизації текстів. Запропоноване архітектурне рішення обумовлено необхідністю оптимізації витрат ресурсів та їх економії під час використання моделі у релізному середовищі за допомогою розробленого веб-сервісу. Здійснено програмну реалізацію запропонованої технології класифікації. Програмний додаток реалізовано засобами мови програмування Python за допомогою бібліотеки для машинного навчання TensorFlow та комплексної платформи Tensorflow Extended. Серверну частину реалізовано з використанням фреймворку aiohttp. Експериментальне дослідження розробленого класифікатору текстів здійснено з використанням News Category Dataset, що представляє собою багатомовні заголовки текстових новин. Застосування запропонованої технології класифікації характеризується незначним погіршенням показників якості під час зміни мови, що дозволяє розробляти багатомовні моделі без втрати їх продуктивності при зміні мови вхідних даних. Результати тестування підтверджують ефективність наведеного підходу.

ВЕКТОРИЗАЦІЯ БАГАТОМОВНИХ ТЕКСТІВ, МОДЕЛЬ XLM-RoBerta, НЕЙРОМЕРЕЖЕВИЙ КЛАСИФІКАТОР БАГАТОМОВНИХ ТЕКСТІВ, МЕРЕЖА LSTM, СЕРВЕР ОБРОБКИ ЗАПИТІВ

Дудник М.П., Удовенко С.Г., Чалая Л.Э., Соколовская М.М. Нейросетевая технология многоязычной классификации электронных текстов. Статья посвящена разработке технологии построения многоязычных классификаторов, основанной на нейросетевой обработке векторного представления текстов, сгенерированного с помощью модели XLM-RoBerta. Рассмотрены преимущества использования векторизации текстов рекуррентной нейронной сети на основе трансформера модели XLM-RoBerta. Приведена схема взаимодействия разработанного классификатора на основе сети LSTM с моделью векторизации текстов. Предложенное архитектурное решение обусловлено необходимостью оптимизации затрат ресурсов и их экономии при использовании модели в релизной среде с помощью разработанного веб-сервиса. Осуществлена программная реализация предлагаемой технологии классификации. Программное приложение реализовано на языке программирования Python с помощью библиотеки для машинного обучения TensorFlow и комплексной платформы Tensorflow Extended. Серверная часть реализована с использованием фреймворка aiohttp. Экспериментальное исследование разработанного классификатора текстов проведено с использованием News Category Dataset, представляющего собой многоязычные заголовки текстовых новостей. Применение предлагаемой технологии классификации характеризуется незначительным ухудшением показателей качества при изменении языка, что позволяет разрабатывать многоязычные модели без потери производительности при изменении языка входных данных. Результаты тестирования подтверждают эффективность приведенного подхода.

ВЕКТОРИЗАЦИЯ МНОГОЯЗЫЧНЫХ ТЕКСТОВ, МОДЕЛЬ XLM-RoBerta, НЕЙРОСЕТЕВЫЙ КЛАССИФИКАТОР МНОГОЯЗЫЧНЫХ ТЕКСТОВ, СЕТЬ LSTM, СЕРВЕР ОБРАБОТКИ ЗАПРОСОВ

Dudnyk M., Udovenko S., Chala L., Sokolovska M. Neural network technology for multilingual classification of electronic texts. The article is devoted to the development of a technology for building multilingual classifiers based on neural network processing of a vector representation of texts generated using the XLM-RoBerta model. The advantages of using the text vectorization of a recurrent neural network based on the transformer of the XLM-RoBerta model are considered. The interaction scheme of the developed classifier based on the LSTM network with the text vectorization model is presented. The proposed architectural solution is due to the need to optimize the cost of resources and save them when using the model in the release environment using the developed web service. The software implementation

of the proposed classification technology has been implemented. The software application is implemented in the Python programming language using the TensorFlow machine learning library and the Tensorflow Extended integrated platform. The server part is implemented using the aiohttp framework. An experimental study of the developed text classifier was carried out using the News Category Dataset, which is a multilingual heading of text news. The use of the proposed classification technology is characterized by a slight deterioration in quality indicators when changing the language, which allows developing multilingual models without loss of performance when changing the language of the input data. The test results confirm the effectiveness of the above approach.

VECTORIZATION OF MULTILINGUAL TEXTS, XLM-RoBerta MODEL, NEURAL NETWORK CLASSIFIER OF MULTILINGUAL TEXTS, LSTM NETWORK, QUERY PROCESSING SERVER

Вступ

В області автоматичної обробки електронних текстів набув розвитку напрямок інтелектуального аналізу даних Text Mining, в рамках якого здійснюється пошук нових корисних знань з неструктурованої текстової інформації. Під неструктурованими текстовими даними розуміють електронні документи будь-якого типу: Web-сторінки, електронну пошту, нормативні документи тощо [1, 2]. Завдання організації ефективного доступу до неструктурованої тематичної інформації безпосередньо пов'язано із завданням класифікації електронних текстів, які надходять з ресурсів мережі Інтернет. Для його вирішення розроблено чимало ефективних методів, деякі з яких характеризуються якістю класифікації, що можна порівняти з результатами класифікації текстів, що виконується кваліфікованими експертами. Під час вирішення задачі автоматичної класифікації текстів використовується їх представлення у векторному вигляді. Існує чимало методів векторизації тексту (зокрема, Word2Vec, TF-IDF, Skip-gram, Continuous Bag of Words, MUSE тощо). Але всі ці методи не дозволяють вирішити задачу багатомовної векторизації текстів, адже їх використання передбачає необхідність попереднього визначення мови, якою написано текст, перед опрацюванням тексту з застосуванням відповідної мови моделі. Однак такий спосіб потребує написання значної кількості коду для підтримки цієї логіки, а також наявності моделі для векторизації та для обробки векторів під кожну мову, з якою потрібно працювати. Розглянемо докладніше модель MUSE (Universal Multilingual Sentence Encoding), що використовується для вирішення проблеми багатомовної класифікації текстів, підтримує 16 мов, та працює на рівні векторизації речення [3]. Основними компонентами цієї моделі є токенізатор тексту «Sentencepiece» та енкодер, який векторизує речення за допомогою архітектури «Transformer». Процес токенізації полягає в перетворенні речення на набір токенів (слів та окремих символів). Особливістю роботи токенізатора, що використовується в моделі MUSE, є можливість опрацьовувати слова, які не присутні в його словнику. Якщо слово не знайдено в словнику токенізатора, то він починає розділяти слово на частини, поки не отримає слова зі словника або окремі букви. Після цього токени заміняються на індекси елементів словнику моделі, що відповідають цим токенам, та подаються на вхід енкодеру, який застосовує двонаправлений Self-attention механізм для обчислення контексту використання токена у реченні. Після цього контекст використання кожного токену у реченні підсумовується та усереднюється для подальшої векторної репрезентації речення. Зазначимо, що розмір вихідного вектору буде обмежений (кількість токенів які можна подати на вхід до блоку «Transformer» дорівнює 512). Таким чином отримується контекстуально зважена векторна репрезентація тексту, що дозволяє опрацювувати тексти на одній з шістнадцяти мов. В той же час, побудувати систему, яка буде одночасно підтримувати, наприклад, як англійську так і українську мову, за допомогою даного методу неможливо. Більш прийнятними для багатомовної реалізації класифікаторів тексту є моделі mBERT та XLM-RoBerta, що передбачають використання замаскованих токенів [4]. На вхід цих моделей подається текст, в якому деякі токени замінені на спеціальний токен-маску, після чого необхідно, користуючись контекстом незамаскованих токенів, визначити які токени знаходяться під маскою. Для багатомовних реалізацій таких моделей можна використовувати навчальний набір даних, де присутні фрагменти текстів кількома мовами. Втім архітектурно багатомовні варіанти моделей mBERT та XLM-RoBerta не відрізняються від їх одномовної реалізації. Для багатомовної моделі необхідно мати датасет, який враховує велику кількість мов. При цьому виникають суттєві проблеми з мовами, що недостатньо представлені в навчальному корпусі. Акуальною є проблема комбінованого використання можливостей моделей типу XLM-RoBerta та рекурентних нейронних мереж для побудови ефективних багатомовних класифікаторів текстів.

У даній роботі пропонується технологія побудови таких класифікаторів, яка основана на нейромережевій обробці векторного подання текстів, згенерованого за допомогою XLM-RoBERTa.

1. Технологія векторизації багатомовних текстів на основі моделі XLM-RoBerta

Для попередньої векторизації текстів в класифікаторах традиційно використовується метод, запропонований в [5], сутність якого полягає в тому що вже для навчених векторних представлень тексту з FastText здійснюється проекція всіх мов у векторний

простір англійської мови (наприклад, вектор одного слова з української мови відображається в аналог цього слова в англійській мові).

Матриця-проектор при цьому обирається так, щоб мінімізувати відстань між словом x_i і його еквівалентною проекцією y_i. Тобто, якщо словник складається з пар (x_i, y_i), то матриця-проектор M формується таким чином:

$$M = argmin_w \sum_i \left\| x_i - Wy_i \right\|^2 , \quad (1)$$

де $\|^2$ – позначення норми.

Крім того, має бути обмежена матриця W проектора, щоб збереглися вихідні відстані між векторами вбудовування слів.

Недоліком такого підходу є те, що на виході отримується векторне подання речення без врахування контексту використання слів та без урахування взаємодії слів між собою в реченні.

Розглянемо переваги використання для векторизації текстів рекуррентної нейронної мережі на основі трансформеру моделі XLM-RoBerta, що підтримує 100 мов. На відміну від традиційних моделей ця модель має вищу точність на великому спектрі задач обробки природно мовних текстів (NLP – Natural-language processing).

Слід відзначити, що моделі mBERT та MUSE також здійснюють векторизацію на рівні речення, як і XLM-RoBerta, але XLM-RoBerta навчена на потужному датасеті CommonCrawl, що складається з 7.8 терабайт текстів англійською мовою та має спільний словник для усіх мов цієї моделі. Це дозволяє збільшити можливість використання проекції у векторному просторі одного й того ж слова різними мовами, що, в свою чергу, дає можливість навчати нейронні мережі векторизації та класифікації однією мовою, а підтримувати всі 100 мов моделі XLM-RoBerta.

Однією з особливостей трансформеру моделі XLM-RoBerta є використання в механізмі внутрішньої уваги (self-attention) технології multi-head attention (багатоголової уваги) [6]. Ця технологія дозволяє замість використання однієї голови уваги з 512 розмірними векторами ключів, запитів та значень лінійно проеціювати ключі, значення та запити h разів з різними вивченими проекціями на відповідні вектори. Якщо значення h дорівнює 8, то кожна голова уваги має розмірність вихідної послідовності 64, що в сумі дозволяє обробляти послідовності, що не перевищують розмірність в 512 токенів. На кожну з цих проекцій ключів, запитів та значень паралельно застосовується функція внутрішньої уваги та отримуються відповідні вектори. Після цього вектори об'єднуються та знову проеціюються, в результаті чого формуються остаточні значення (рис. 1).

Багатоголова увага дозволяє моделі XLM-RoBerta здійснювати доступ паралельно до різних частин вхідного речення. Формула для обчислення багатоголової уваги має наступний вигляд:

$$MultiHead(Q,K,V) = Concat(head_1, \ldots, head_h)W^O \ , (2)$$

де $head_1 = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right)$; W_i^Q – матриця ваг для матриці запитів i; W_i^K – матриця ваг для матриці ключів i; W_i^V – матриця ваг для матриці значень i.

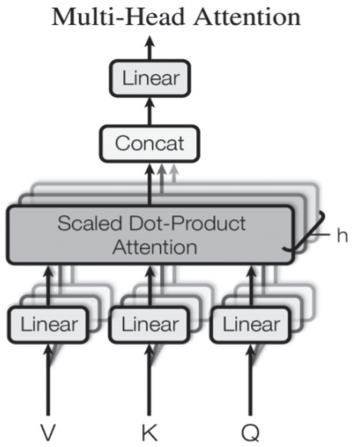


Рис. 1. Ілюстрація механізму багатоголової уваги моделі XLM-RoBerta

Результати порівняння складності операцій в різних шарах нейронної мережі з механізмом внутрішньої уваги, що передбачається далі використовувати у багатомовному класифікаторі, наведено в табл. 1.

Таблиця 1

Результати порівняння складності операцій в різних шарах нейронної мережі

Тип шару	Складність шару	Послідовність операцій	Максимальна довжина
Внутрішня увага	$O(n^2 * d)$	$O(1)$	O(1)
Рекурентний	$O(n * d^2)$	$O(n)$	O(n)
Згортковий	$O(k * n * d^2)$	$O(1)$	$O(log_k(n))$
Обмежена внутрішня увага	$O(r * n * d)$	$O(1)$	$O(n/r)$

У табл. 1 прийнято такі позначення: n – кількість елементів у вхідній послідовності; d – розмір простору репрезентації вхідних даних (розмірність словнику моделі); k – розмір вікна операції згортки, r – кількість сусідів в шарі обмеженої складності внутрішньої уваги (Self-Attention restricted).

Результати порівняння показують, що шар внутрішньої уваги з'єднує усі позиції з постійною кількістю послідовно виконаних операцій, в той час рекурентна мережа потребує $O(n)$ послідовних операцій. Шар власної уваги працює швидше, ніж рекурентна мережа, коли вхідна послідовність n менша ніж розмір простору репрезентації d, що є досить поширеною ситуацією.

Якщо замість рекурентних мереж використовувати згорткову мережу з розміром вікна $k < n$, то виникає проблема з'єднання вхідних та вихідних позицій, адже декодер буде втрачати частину інформації, що надійшла з енкодеру. Щоб вирішити цю проблему, потрібно використання $O(n/k)$ згорткових мереж. При цьому необхідним є врахування позиції кожного слова у вхідних даних моделі. Позиційне кодування додається до вхідних ембедінгів слів у нижніх частинах стеків енкодерів та декодерів. Так як позиційні кодування мають розмірність, яка співпадає з розмірністю ембедінгу, то позиційне кодування додають до ембедінгів. Доцільним є використання синусоїдальної функції позиційного кодування:

$$PE_{(pos,2i)} = sin\left(pos / 10000^{2i/d_{model}}\right), \qquad (3)$$

де pos – позиція елементу у вхідній послідовності; i – розмірність.

Кожний вимір позиційного кодування відповідає синусоїді. Довжини хвиль утворюють геометричну прогресію від 2π до $10000 \cdot 2\pi$. Така функція дозволяє моделі обробляти послідовності, що мають більшу довжину відносно послідовностей в тренувальних даних, так як для будь якого фіксованого зсуву k функція $P_{\text{pos}} + k$ може бути представлена як лінійна функція від P_{pos}.

В трансформері моделі XLM-RoBerta, кожна компонента енкодера має залишковий зв'язок навколо себе, за котрим слідує етап нормалізації шару (layer-normalization step). Візуалізацію операції нормалізації шару внутрішньої уваги наведено на рис. 2 [7].

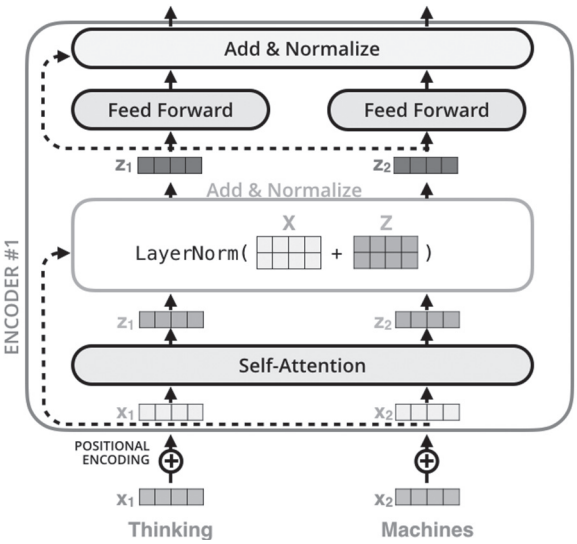


Рис. 2. Візуалізація операції нормалізації шару внутрішньої уваги моделі XLM-RoBerta

На етапі регуляризації в трансформері моделі XLM-RoBerta до вихідних даних кожного елементу шару енкодера та декодера застосовується техніка дропаут (Dropout). Згідно з цією технікою під час навчання на різних ітераціях відбувається вилучення певного проценту нейронів (як у прихованих шарах так і у видимих). Внаслідок цього найбільш натреновані нейрони отримують найбільші ваги. Крім того, дропаут використовується для суми ембедінгів з позиційним кодуванням (як в енкодері так і в декодері) з коефіцієнтом 0.1, тобто 10 відсотків цієї суми відкидається. Блок декодеру трансформеру моделі XLM-RoBerta подібний за архітектурою з блоком енкодеру, але має додатковий шар – encoder-decoder attention. Внутрішньо робота декодеру не відрізняється від роботи енкодеру, за винятком нюансів маскування усіх токенів, що йдуть після поточного токена [8].

Після того, як вхідна послідовність пройде через усі блоки енкодерів, з вихідних даних останнього енкодеру формуються матриці ключів та значень, які використовуються шаром encoder-decoder attention. При цьому encoder-decoder attention працює як механізм багатоголової уваги, за винятком того, що він створює матрицю запиту з попереднього шару, який знаходиться нижче, та отримує матриці значень та ключів з вихідних даних стеку енкодерів.

Внутрішня увага у шарі декодеру може фокусуватися лише на попередніх позиціях відносно поточної. Це потрібно для збереження можливості авторегресії та досягається за допомогою маскування всіх наступних токенів відносно поточного. Значення для замаскованих токенів встановлюються перед етапом софтмакс в обчисленні внутрішньої уваги.

Після того як декодер завершує свою роботу, його вихід подається до повнозв'язної мережі, яка формує репрезентацію вихідних даних моделей. Тобто її вихід буде складатися з вектору, який матиме таку ж саму розмірність, як і словник моделі, а кожний елемент вектору репрезентує кожне слово, що є в словнику відносно вхідних даних. Далі цей вектор подається на функцію софтмакс, яка перетворює репрезентацію кожного слова на його вірогідність, після чого формується вектор з вірогідністю кожного слова. Відібравши індекси елементів з найбільшою вірогідністю, отримаємо індекси слів у словнику, з яких і складається вихідне речення.

2. Класифікатор багатомовних текстів на основі моделі XLM-RoBERTa та мережі LSTM

Для вирішення задачі багатомовної класифікації тексту було розроблено нейромережевий класифікатор, що обробляє векторне подання, згенероване за допомогою моделі XLM-RoBerta. Схему взаємодії цього класифікатору з моделлю XLM-RoBerta наведено на рис. 3.

XLM-RoBerta в даному класифікаторі знаходиться зовні архітектури та є окремою моделлю, яка не приймає участь в навчанні. Запропоноване архітектурне рішення обумовлено необхідністю оптимізації витрат ресурсів та їх економії під час використання моделі у релізному середовищі за допомогою розробленого

веб- сервісу. Після того, як сервер отримує запит від клієнта на обробку тексту, він посилає запит до Tensorflow Serving Server, вже підготовлений для моделі XLM-RoBerta вхідний текст та отримує тензор як результат виконання запиту, після чого подає цей тензор знову на сервер з моделями, але вже в якості вхідних даних для моделі класифікації. Після обробки класифікатором даного тензору отримується відповідь у вигляді розподілу вірогідностей кожного класу, що надалі трансформується в текст та надається клієнту, як остаточна відповідь [9].

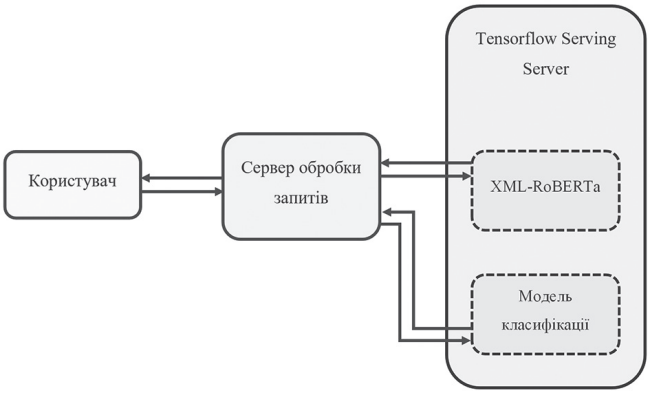


Рис. 3. Схема взаємодії класифікатору з моделлю XLM-RoBerta

Даний підхід дозволяє інваріантно до кількості запитів, що надходять для обробки одного тексту, оптимізувати ресурси та середній час обробки одного запиту клієнта. Завдяки тому, що моделі знаходяться на окремому сервері, при клонуванні серверу обробки запитів потрібно менше ресурсів, тому що моделі не клонуються разом з ним. Клонування може бути потрібно, якщо працює балансер навантаження та підіймає додаткові віртуальні машини з веб-сервісом при зростанні навантаження.

Модель класифікатору складається з 4 шарів. Перший шар (InputLayer) є стандартним шаром, що отримує дані та передає їх далі у модель. Шар LSTM – основний шар моделі, що аналізує вектор, який надходить від XLM-RoBerta. Шари Dense – останні два шари мережі, перший з яких виконує функцію зниження розмірності та має функцію активації ReLU, а другий є софтмакс шаром, який відповідає за класифікацію тексту.

Схематичне зображення моделі нейромережевого класифікатору наведено на рис. 4.

Встановлення платформи Tensorflow Serving на сервер та її компілювання з підтримкою інструкцій CPU (таких як AVX, SSE та інші) дозволяють отримати сервер хостингу моделей, оптимізований конкретно під характеристики серверу, що використовується. Використання платформою Tensorflow Serving заморожених моделей, які потребують незначних ресурсів оперативної пам'яті, сприяє підвищенню продуктивності роботи серверу та значній економії ресурсів.

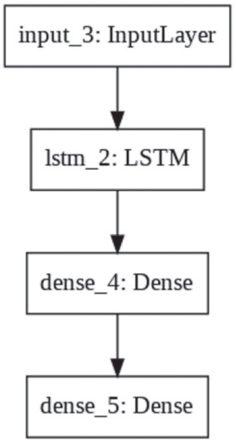


Рис. 4. Узагальнена архітектура пропонованого класифікатору

Розглянемо детальніше особливості архітектури рекуррентної мережі LSTM (Long Short Term Memory), що реалізає завдання класифікації [10]. Ця мережа є модифікацією архітектури рекурентних нейронних мереж RNN, що були розроблені для вирішення задач, в яких вхідна послідовність представлена у вигляді ряду. Але якщо вхідні послідовності мають велику довжину, то архітектура RNN працює незадовільно через проблему затухаючого градієнту: градієнт під час проходження від кінця мережі до її початку становиться занадто малим, внаслідок чого не змінюються ваги шарів в мережі. Вирішенню даної проблеми сприяє архітектура LSTM (рис. 5).

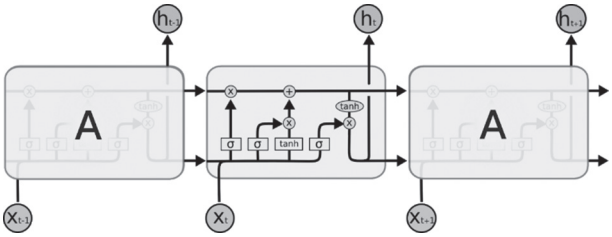


Рис. 5. Узагальнена архітектура мережі LSTM

В LSTM використовуються у кожній комірці вхідний фільтр, вихідний фільтр та фільтр-відсіювач.

Фільтр-відсіювач необхідний для відсіювання нерелевантних даних з попередньої комірки, що допомагає використовувати лише релевантну для нейронної мережі інформацію в поточний час. Вихід даного фільтра формується таким чином:

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f, \quad (4)$$

де W_f – ваги для шару фільтру; h_{t-1} – вихід попереднього шару; x – вхідні дані; b_f – зсув

Фільтр вхідних даних потрібен для того, щоб визначати, яку інформацію з вхідної послідовності потрібно зберегти у стані даної комірки. Вихід даного фільтра формується таким чином:

$$i_t = \sigma\left(W_i * [h_{t-1}, x] + b_i\right), \quad (5)$$

де W_i – ваги для шару фільтру; h_{t-1} – вихід попереднього шару; x – вхідні дані; b_i – зсув.

Фільтр вихідних даних потрібен для коригування інформації, яку ми хочемо передати у наступну комірку. Вихід даного фільтра формується таким чином:

$$o_t = \sigma\left(W_o * \left[h_{t-1}, x\right] + b_o\right), \quad (6)$$

де W_o – ваги для шару фільтру; h_{t-1} – вихід попереднього шару; x – вхідні дані; b_o – зсув.

LSTM здатна обробляти великі послідовності текстових даних без виникнення проблем з вагами мережі, які не змінюються.

Згідно зі схемою взаємодії кінцевого клієнту з додатком, що наведена на рис. 3, необхідно розробити модуль для звернення до серверу з моделями Tensorflow Serving та методи обробки запиту від клієнту. Платформа Tensorflow Serving характеризується наявністю високорівневого API Keras, що дозволяє зменшити кількість часу, необхідного для створення прототипів моделей, та зменшує складність коду, необхідного для створення програмного продукту.

3. Експериментальні дослідження

Розглянемо результати експериментальних досліджень розробленого нейромережевого класифікатору багатомовних текстів, що обробляє векторне подання, згенероване за допомогою моделі XLM-RoBerta.

Для тренування класифікатору використовувався датасет, що був завантажений з відкритої платформи kaggle.com, призначеної для здійснення машинного навчання, яка має відносно великий репозиторій публічних датасетів для різних завдань.

Датасет, що було обрано для експериментального дослідження розробленого класифікатору текстів, має назву News Category Dataset та представляє собою багатомовні заголовки текстових новин [11]. Вибір лише заголовків для тестування обумовлюється тим, що архітектура трансформеру моделі XLM-RoBerta має обмеження на довжину вхідної послідовності в 512 токенів, а текст новин зазвичай перевищує задану максимальну довжину послідовності.

Структура датасету відповідає формату JSON та містить наступні реквізити: Category (визначає категорію, до якої відноситься текст); Headline (заголовок статті); authors (автори статті); link (посилання на статтю), short_description (короткий описа теми статті), date (дата публікації). Оригінальний датасет містить 40 різноманітних категорій, зібраних з HuffPost.

Для виконання експериментального моделювання та навчання моделі для подальшого використання в системі датасет було оброблено та скорочено до 5 категорій, кожна з яких містить 500 записів. Таким чином датасет було збалансовано.

Фрагмент обробленого фінального датасету наведено на рис. 6.

text	label
Dean Foods Ex-Chairman, Pro Gambler Charged Wi...	SPORTS
Royal Rumble Winner Rumors! Will Kenny Omega M...	SPORTS
LeBron James Flopped So Hard That 'LMAO LeBron...	SPORTS
We Need These Old-School Baseball Cards Featur...	SPORTS
D.C. United 2 - 1 Philadelphia Union...but the...	SPORTS
...	...
Monday Matters: A Walk Home To Remember, An Un...	GOOD NEWS
Refuge in Americal remember his brown leather ...	GOOD NEWS
Dog Discovers The Many Wonders Of A Hula HoopF...	GOOD NEWS
Teacher And His Students Recreate 'Uptown Funk...	GOOD NEWS
Man Has Adorable Conversation With Dozens Of B...	GOOD NEWS

Рис. 6. Фрагмент обробленого датасету

Поле text представляє собою сумму компонент headline та short_description, а поле label – категорію, до якої відноситься даний текст. Усі перетворення були виконані за допомогою можливостей, що надає бібліотека Pandas. Даний датасет представляє собою навчальну вибірку для дослідження розробленого класифікатору.

Для валідаційного датасету було відібрано 10% від навчального та переведено за допомогою translate-api на іспанську мову.

Для проведення експериментів з навчання моделей було використано середовище Google Collaboratory, що надає 8 гб GPU TESLA K80. Під час моделювання досліджувалися моделі XLM-RoBerta base та mBERT base.

В моделях класифікації використовувалася векторна репрезентація тексту, згенерована за допомогою трансформерів mBERT та XLM-RoBerta, що надається відкритим хабом моделей від huggingface. Завантаження моделі в середовище здійснювалося наступним чином:

```
tokenizer=BertTokenizer.from_pretrained('bert-base-multilingual-cased')
transformer_mode=TFBertModel.from_pretrained('bert-base-multilingual-cased').
```

Так як класифікатор використовує вже готову векторну репрезентацію, наступним кроком була генерація ембедінгів, що здійснювалася згідно з наступним кодом:

```
def generate_embeddings(texts, model, tokenizer):
list_of_embeddings = []
for text in tqdm(texts):
tokenizer_output = [to_tokens(text, tokenizer)]
input_ids_train = np.array(select_field(tokenizer_output, 'input_ids'))
attention_masks_train = np.array(select_field(tokenizer_output, 'attention_mask'))
inputs = [input_ids_train,
```

```
attention_masks_train]
embeddings = model.predict(inputs)
# print(np.shape(embeddings[0]))
list_of_embeddings.append(embeddings[0][0])
return np.array(list_of_embeddings).
```

Після підготовки даних здійснювалися генерація моделі класифікатору та запуск навчання згідно з наступним кодом:

```
def create_model():
embeddings = tf.keras.layers.Input((125,768), dtype=
tf.float32)
x=tf.keras.layers.LSTM(units=512,dropout=0.2,recur
rent_dropout=0.2)(embeddings)
x = tf.keras.layers.Dense(units=256, activation='relu')
(x)
outputs = tf.keras.layers.Dense(5, activation=
'softmax')(x)
model = tf.keras.models.Model(inputs=embeddings,
outputs=outputs)
return model
model = create_model()
model.summary()
model.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
history = model.fit(embeddings, y_train, batch_size=
12, epochs=10).
```

В якості функції похибки було обрано стандартну для завдань багатокласової класифікації категорійну кросс-ентропію.

Після завершення навчання класифікатору з трансформером mBERT на тренувальному наборі даних, яке відбувалось 10 епох, були отримали наступні результати: метрика точності (accuracy) дорівнювала 0.9711, а значення функції похибки дорівнювало 0.0767. Зміну значень функції похибки та accuracy для моделі з трансформером mBERT base наведено на рис. 7 та рис. 8.

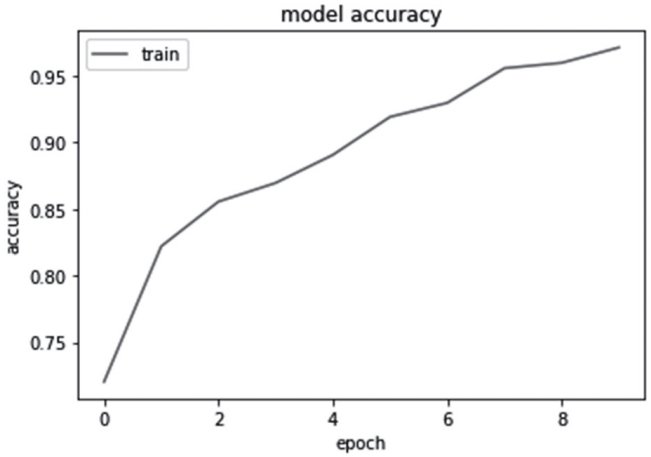


Рис. 7. Зміна метрики accuracy для моделі з трансформером mBERT base

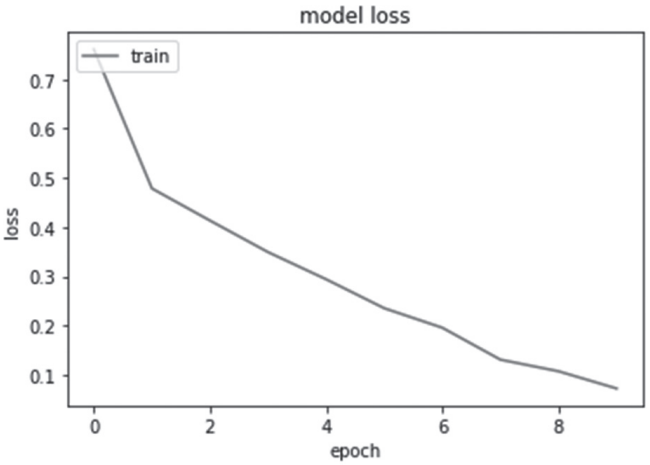


Рис. 8. Зміна функції похибки для моделі з трансформером mBERT base

Наступним кроком була генерація векторного подання для валідаційного набору даних та підрахунок функції похибки та метрики accuracy для валідаційного набору даних згідно з таким кодом:

```
eval_history=model.evaluate(eval_embeddings, y_test,
batch_size=2).
```

Значення функції похибки після закінчення підрахунку дорівнювало 1.6751, а метрики accuracy 0.7400. З цього можна зробити наступний висновок: моделі на базі mBERT показують прийнятний результат для англійської мови, але мають спад точності в разі зміни мови тексту.

Після завершення навчання класифікатору з трансформером XLM-RoBerta на тренувальному наборі даних, яке відбувалось 10 епох, були отримані наступні результати: метрика точності (accuracy) дорівнювала 0.8732, а значення функції похибки дорівнювало 0.4020. Зміну значень функції похибки та accuracy для моделі з трансформером XLM-RoBerta наведено на рис. 9 та рис. 10.

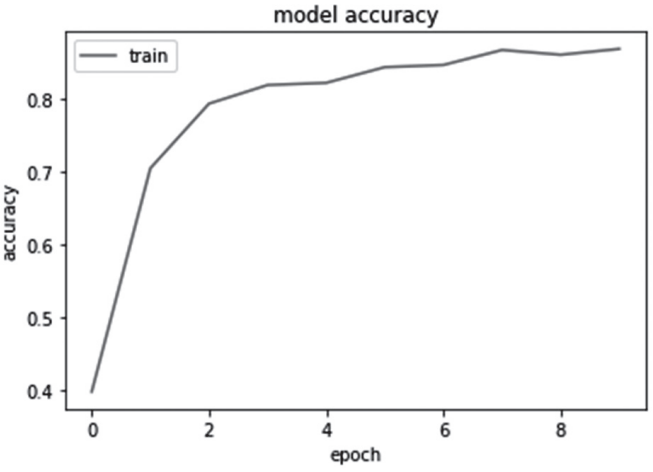


Рис. 9. Зміна метрики accuracy для моделі з трансформером XLM-RoBerta

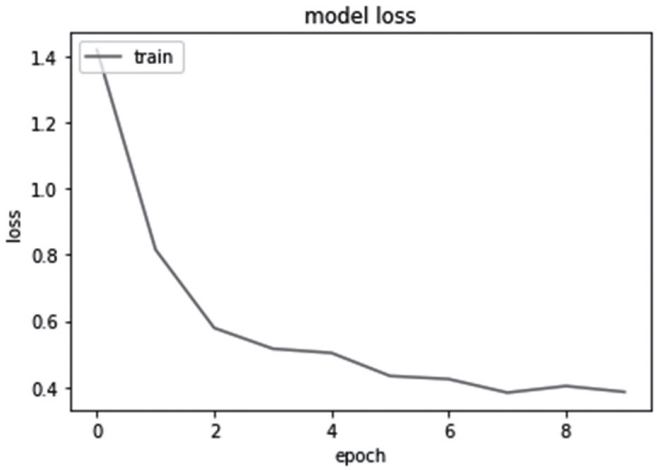


Рис. 10. Зміна функції похибки для моделі з трансформером XLM-RoBerta

Після підрахунків відповідних показників для валідаційного набору даних метрика accuracy дорівнювала 0.81, а значення функції похибки дорівнювало 0.5030. З цього можна зробити висновок, що XLM-RoBerta на відміну від mBERT має незначне погіршення метрик під час зміни мови, що дозволяє розробляти багатомовні моделі без втрати продуктивності моделі при зміні мови вхідних даних.

За результатами експериментального моделювання було обрано модель XLM-RoBerta для подальшої розробки програмного додатку, що реалізує процедури звернення користувача до серверу класифікатору та обробку відповідних запитів.

4. Розробка програмного додатку

Згідно зі схемою взаємодії запропонованого багатомовного класифікатору з моделлю XLM-RoBerta (рис. 3) були розроблені програмні модулі для звернення користувача до серверу з моделями Tensorflow Serving та методи обробки відповідних запитів. Для модуля з посиланням запитів до Tensorflow Serving та їх подальшою обробки були розроблені класи CustomEmbGenerator та TextClassifier.

CustomEmbGenerator відповідає за звернення до моделі XLM-RoBerta для генерації ембедингів та реалізує методи для токенізації тексту, його форматування для вхідного формату моделі та посилання запиту до моделі через проток grpc. Відправка запитів до моделі XLM-RoBerta відбувається згідно з наступним кодом:

```
def _send_request(self, body: List[‘np.ndarray’]) -> ‘np.ndarray’:
    Send gRPC request to model.
    :param body: Request body.
    :return: Embeddings.
    ``````
    logger = logging.getLogger(‘generator.serving’)
    serving = ServingGRPCClient(self.address, None)
    try:
    logger.info(‘Sending gRPC request to XLM-R model ...’)
    result = serving.predict(name=»xlm-roberta-base»,
    inputs={‘input_ids’: tf.compat.v1.make_tensor_proto (body[0]),
    ‘attention_mask’: tf.compat.v1.make_tensor_proto (body[1])})
    embeddings = np.squeeze(tf.make_ndarray(result. outputs[‘output_0’]))
    except Exception as exception: # pylint: disable=broad-except
    embeddings = []
    logger.exception(exception)
    return embeddings.
```

Як можна бачити, відправлення запиту до віддаленого серверу з використанням grpc дещо відрізняється від класичного протоколу HTTP. Наступним кроком була розробка класу TextClassifier для звернення та обробки результатів роботи моделі класифікації. Даний клас реалізує два методи: send_grpc_request та analyze. Перший метод має вихідний код, що є подібний до методу з класу CustomEmbGenerator. Як вхідні дані цей метод приймає результат, отриманий від моделі XLM-RoBerta та посилає його до моделі класифікації для подальшої обробки, а у відповідь отримує тензор розподілу вірогідностей кожного класу, згідно з яким натренована модель класифікації. Метод analyze визиває методи класу CustomEmbGenerator для отримання ембедінгів та передає їх у метод send_grpc_request для отримання відповіді від моделі класифікації, а потім перетворює вірогідності на класи, відповідно словнику. Метод працює згідно з наступним кодом:

```
def analyze(self, input_data: Any) -> Tuple:
    ``````
    Analyzing sentiments.
    If  input data is str that contains a text returns list that contain a np.array with sentiments score.
    :param input_data: string or list of strings.
    :return: ``np.ndarray`` with sentiments score if single text or
    ``List[np.ndarray]`` with sentiments score if multiple.
    ``````
    logger = logging.getLogger(‘classifier.analyze’)
    emb_gen = CustomEmbGenerator(max_pad_length=125, crop=False)
    # print(input_data)
    logger.debug(«IN ANALYZE»)
    if isinstance(input_data, list):
    count_of_sentences = []
    sentence_list = []
    all_sentence_list = []
    logger.debug(input_data)
    # print(input_data)
```
```

```
for i, text in enumerate(input_data):
sentences = sent_tokenizer(text)
sentence_list.append(sentences)
 count_of_sentences.append(len(sentences))
all_sentence_list.append(list(chain(*sentence_
list)) logger.debug(f'All sentences in input data array: {all_
sentence_list}')
logger.debug('Creating embeddings for texts')
embeddings = emb_gen.create_embedding(all_
sentence_list[0], batching=True, return_full=True)
prediction = self._send_grpc_request(embeddings)
predictions = emb_gen.chunk_data(prediction,
count_of_sentences)
labels = [self.__labels[p.argmax()] if float(p[p.
argmax()]) > 0 else 0 for p in predictions]
# probability for label class
probabilities = [p[p.argmax()] for p in predictions]
else:
sentence_list = sent_tokenizer(input_data)
logger.debug(f»Sentence list: {sentence_list}»)
if len(sentence_list) == 1:
embeddings = [emb_gen.create_embedding(sentence_
list[0], return_full=True)]
else:
embeddings = emb_gen.create_embedding(sentence_
list, return_full=True)
prediction = self._send_grpc_request(embeddings)
prediction = prediction.astype(float)
if prediction.ndim == 2:
labels = [self.__labels[p.argmax()] if float(p[p.
argmax()]) > 0 else 0 for p in prediction]
probabilities = [p[p.argmax()] for p in prediction]
else:
labels = [self.__labels[prediction.argmax()] if float
(prediction.argmax()) > 0.2 else 0]
# probability for label class
probabilities = [prediction[prediction.argmax()]]
return labels, probabilities.
```

Для обробки запиту від клієнту було розроблено хендлер (спеціальний метод, який дозволяє обробляти HTTP запити), що використовує aiohttp. Розроблений хендлер реалізує метод з назвою text_analayzer та відповідає на запит, що здійснено за адресою app_url/text_analyze. Для його коректної роботи необхідно, щоб у заголовку запита від клієнту містився ключ доступу. Дана опція дозволяє перевірити, чи має клієнт права доступу до системи.

Для перевірки можливостей додатку в якості клієнту буде використано програмний засіб Postman, що дозволяє тестувати веб-сервіси з можливостю посилання запитів до серверу.

Програмний додаток було реалізовано засобами високорівневої мови програмування Python за допомогою відкритої програмної бібліотеки для машинного навчання TensorFlow та комплексної платформи для розгортання виробничих конвеєрів машинного навчання Tensorflow Extended, серверну частину якого було реалізовано за допомогою фреймворку aiohttp, та за допомогою програмної бібліотеки мовою Python для обробки і аналізу даних Pandas.

Розроблений веб-сервіс здатен вирішувати завдання багатомовної класифікації з прийнятною точністю (зокрема, класифікувати описи новин на 100 мовах).

Перспективним продовженням проведених досліджень можуть бути: інтеграція багатомовного класифікатору з інструментами великих даних; розширення функціональних можливостей сервісу (розробка модулів для емоціонального аналізу тексту, пошуку в текстах подібного контенту тощо).

Висновки

Запропонована технологія сприяє вирішенню завдання багатомовної класифікації природномовних електронних текстів, що надходять з ресурсів мережі Інтернет або електронних бібліотек. Основою цієї технології є комбіноване використання можливостей моделей типу XLM-RoBerta та рекуррентних нейронних мереж типу LSTM для побудови ефективних багатомовних класифікаторів текстів.

Проведений аналіз особливостей векторного подання текстів свідчить, що найбільш прийнятною для багатомовної реалізації класифікаторів тексту є модель XLM-RoBerta, яка передбачає використання замаскованих токенів. Для багатомовних реалізацій цієї моделі можна використовувати навчальний набір даних, де присутні фрагменти текстів кількома мовами. XLM-RoBerta в даному класифікаторі знаходиться зовні архітектури та є окремою моделлю, яка не приймає участь в навчанні. Запропоноване архітектурне рішення обумовлено необхідністю оптимізації витрат ресурсів та їх економії під час використання моделі у релізному середовищі за допомогою розробленого веб-сервісу.

Модель класифікатору реалізовано з використанням чотирьохшарової нейронної мережі, основним компонентом якої є шар LSTM, що аналізує вектор, який надходить від трансформеру моделі XLM-RoBerta. Після того, як сервер класифікатору отримує запит від клієнта на обробку тексту, він посилає запит до Tensorflow Serving Server та отримує тензор як результат виконання запиту, після чого подає цей тензор знову на сервер з моделями, але вже в якості вхідних даних для моделі класифікації. Після обробки класифікатором даного тензору отримується відповідь у вигляді розподілу вірогідностей кожного класу, що надалі трансформується в текст та надається клієнту, як остаточна відповідь.

Експериментальне дослідження розробленого класифікатору текстів здійснено з використанням

News Category Dataset, що представляє собою багатомовні заголовки текстових новин.

Застосування запропонованої технології класифікації характеризується незначним погіршенням показників якості під час зміни мови, що дозволяє розробляти багатомовні моделі без втрати продуктивності моделі при зміні мови вхідних даних.

Можна вважати доцільним продовження досліджень щодо удосконалення розробленого багатомовного класифікатора з метою підвищення його функціональних можливостей та поліпшення якісних характеристик.

Список літератури:

[1] Чалая Л.Э. Метод двухэтапной классификации электронных текстов // Л.Э. Чалая, С.Г. Удовенко, Е.С. Кушвид // Біоніка интелекту. – 2016. – № 2 (87). – С.16 – 23.

[2] Інформаційні технології та системи: монографія / Удовенко С.Г. Розділ 8. Класифікація електронних науково-технічних текстів в інформаційно-пошукових системах // С.Г. Удовенко, Л.Е. Чала. – Х.: ФОП Бровін О.В., 2019. – С.108 – 123.

[3] Yang Y., Cer D., Amin A., Guo M., Law J., Constant N., Hernandez Abrego G., Yuan S., Tar C., Yun-Hsuan S., Strope B., Kurzweil R. Multilingual Universal Sentence Encoder for Semantic Retrieval, 9 Jul, 2019, URL: https://arxiv.org/pdf/1907.04307.pdf (Last accessed: 15.05.2021).

[4] Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 11 Oct, 2018, URL: https://arxiv.org/pdf/1810.04805.pdf (Last accessed: 15.05.2021).

[5] Stoyanov V., Necip F. A., Under the hood: Multilingual embeddings, January 24, 2018, URL: https://ai.facebook.com/blog/under-the-hood-multilingual-embeddings/ (Last accessed: 15.05.2021).

[6] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., Polosukhin I., Attention Is All You Need, 12 Jun, 2017, URL: https://arxiv.org/pdf/1706.03762.pdf (Last accessed: 15.05.2021).

[7] Alammar J. The Illustrated Transformer, 27 Jun, 2018, URL: http://jalammar.github.io/illustrated-transformer/ (Last accessed: 15.05.2021).

[8] Conneau A., Khandelwal K., Goyal N., Chaudhary V., Wenzek G., Guzmán F., Grave E., Ott M., Zettlemoyer L., Stoyanov V. Unsupervised Cross-lingual Representation Learning at Scale, 8 Apr, 2020, URL: https://arxiv.org/pdf/1911.02116.pdf (Last accessed: 15.05.2021).

[9] Tay Y., Dehghani M., Gupta J., Bahri D., Aribandi V., Qin Z., Metzler D. Are Pre-trained Convolutions Better than Pre-trained Transformers? 7 May, 2021, URL: https://arxiv.org/pdf/2105.03322.pdf (Last accessed: 15.05.2021).

[10] Olah C. LSTM - мережі довгої короткострокової пам'яті, 21 червень, 2017, URL: https://habr.com/ru/company/wunderfund/blog/331310/ (Last accessed: 15.05.2021).

[11] Misra R. News Category Dataset, 2 Dec, 2018, URL: https://www.kaggle.com/rmisra/news-category-dataset (Last accessed: 15.05.2021).

Надійшла до редколегії 17.09.2021

УДК 519.7:007.52; 519.711.3 DOi 10.30837/ bi.2021.2(97).02

І.Д. Вечірська[1], М.С. Черноусова[2], А.Д. Вечірська[3]

[1]кандидат технічних наук, доцент кафедри інформатики,
Харківський національний університет радіоелектроніки,
iryna.vechirska@nure.ua, ORCID ID: 0000-0001-7964-2361

[2]студент групи ІТІНФ-19-2,
Харківський національний університет радіоелектроніки,
mariia.chernousova@nure.ua, ORCID ID: 0000-0002-2750-0916

[3]студент групи ІТІНФ-20-3,
Харківський національний університет радіоелектроніки,
anna.vechirska@nure.ua, ORCID ID: 0000-0002-3886-3026

ПОБУДОВА ЛОГІЧНОЇ МЕРЕЖІ ДЛЯ ОПИСУ ПРОЦЕСУ КОРЕСПОНДЕНЦІЇ БАЛАНСОВИХ РАХУНКІВ ОБЛІКУ ДЕПОЗИТІВ СУБ'ЄКТІВ ГОСПОДАРСЬКОЇ ДІЯЛЬНОСТІ ТА ФІЗИЧНИХ ОСІБ

Статтю присвячено дослідженню інструментарія алгебри скінченних предикатів для формалізації процесів бухгалтерського обліку, зокрема процесу кореспонденції балансових рахунків обліку депозитів суб'єктів господарської діяльності та фізичних осіб. Бухгалтерський облік банку є складовою інформаційної системи банку та включає облік операцій за відповідними рахунками на підставі автоматизованих та ручних проводок, складання агрегованих та детальних звітів. План рахунків бухгалтерського обліку банків України – це систематизований перелік рахунків бухгалтерського обліку, який використовується для детальної та повної реєстрації всіх банківських операцій. Введення предикату впізнавання предметів на вказаній предметній області дозволило формально описати дані будь-якого типу, а застосований метод побудови логічних мереж забезпечує підвищення швидкості обробки інформації за рахунок розпаралелювання процесів обробки. Таким чином, складне багатомісцеве відношення було розбито на композицію бінарних відношень, що описуються мовою алгебри предикатів з урахуванням деталізованого системного аналізу предметної області. Математична модель процесу кореспонденції рахунків обліку депозитів суб'єктів господарської діяльності та фізичних осіб подано предикатом, що залежить від одинадцяти змінних. Цей предикат характеризується системою одинадцяти бінарних відношень, які в статті представлено дводольними графами та формулами відповідних предикатів. Предикат моделі має вигляд кон'юнкції усіх побудованих вище бінарних предикатів. Побудована логічна мережа процесу кореспонденції балансових рахунків працює ітераційно до тих пір, поки не отримає сталі результати на двох кроках підряд та дозволяє розв'язувати задачі аналізу та синтезу інформаційної системи банку та включає облік операцій за відповідними рахунками.

ПРЕДИКАТ, ЛОГІЧНА МЕРЕЖА, АЛГЕБРА СКІНЧЕННИХ ПРЕДИКАТІВ, БАЛАНСОВИЙ РАХУНОК, АНАЛІТИЧНИЙ ОБЛІК, ДЕБІТ, КРЕДІТ

Вечирская И.Д., Черноусова М.С., Вечирская А.Д. Построение логической сети для описания процесса корреспонденции балансовых счетов учета депозитов субъектов хозяйственной деятельности и физических лиц. Статья посвящена исследованию инструментария алгебры конечных предикатов для формализации процессов бухгалтерского учета, в частности процесса корреспонденции балансовых счетов учета депозитов субъектов хозяйственной деятельности и физических лиц. Бухгалтерский учет банка является составляющей информационной системы банка и включает учет операций по соответствующим счетам на основании автоматизированных и ручных проводок, составления агрегированных и подробных отчетов. План счетов бухгалтерского учета банков Украины – это систематизированный перечень счетов бухгалтерского учета, который используется для подробной и полной регистрации всех банковских операций. Введение предиката узнавания предмета на указанной предметной области позволило формально описать данные произвольного типа, а примененный метод построения логических сетей обеспечивает повышение скорости обработки информации за счет распараллеливания процессов обработки. Таким образом, сложное многоместное отношение было разбито на композицию бинарных отношений, которые описываются языком алгебры конечных предикатов с учетом детализированного системного анализа предметной области. Математическая модель процесса корреспонденции счетов учета депозитов субъектов хозяйственной деятельности и физических лиц подано предикатом, который зависит от одиннадцати переменных. Этот предикат характеризируется системой одиннадцяти бинарных отношений, которые в статье представлены двудольными графами и формулами соответствующих предикатов. Предикат модели имеет вид конъюнкции всех построенных выше бинарных предикатов. Построенная логическая сеть процесса корреспонденции балансовых счетов работает итерационно до тех пор, пока не получит постоянные результаты на двух шагах подряд и позволяет решать задачи анализа и синтеза информационной системы банка, включая учет операций по соответствующим счетам.

ПРЕДИКАТ, ЛОГИЧЕСКАЯ СЕТЬ, АЛГЕБРА КОНЕЧНЫХ ПРЕДИКАТОВ, БАЛАНСОВЫЙ СЧЕТ, АНАЛИТИЧЕСКИЙ УЧЕТ, ДЕБИТ, КРЕДИТ

Vechirska I.D., Chernousova M.S., Vechirska A.D. Creating logical network to describe the process of correspondence of balance sheet accounts of accounting for deposits of business entities and individuals. The article is devoted to the study the toolkit of the algebra of finite predicates for the formalization of accounting processes, in particular, the process of correspondence of balance sheet accounts of accounting for the deposits of business entities and individuals. The bank's accounting is a component of the bank's information system and includes accounting of transactions on the relevant accounts based on automated and manual postings, compilation of aggregate and detailed reports. The chart of accounts of bank's accounting of Ukraine is a systematized list of accounts of accounting, which is used for detailed and complete registration of all banking operations. The introduction of the predicate for recognizing an object in the specified area made it possible to describe arbitrary type data formally. Applied method of creating logical networks provides the increase in the speed of information processing due to parallelization of processing. Thus, a complex multiplace relation was transformed into a composition of binary relations, which are described in the language of finite predicate algebra, considering a detailed system analysis of the subject area. The mathematical model of the process of correspondence of accounts of deposits of business entities and individuals is given by a predicate that depends on eleven variables. This predicate is characterized by a system of eleven binary relations, which are represented in the article by bipartite graphs and formulas of the corresponding predicates. The model predicate has the form of a conjunction of all the binary predicates built above. The constructed logical network of the correspondence process of balance sheet runs iteratively until get constant results on two steps in a row and accounts allows solving the problems of analysis and synthesis of the bank's information system, including accounting of transactions on the corresponding accounts.

PREDICATE, LOGICAL NETWORK, FINITE PREDICATE ALGEBRA, BALANCE ACCOUNT, ANALYTICAL ACCOUNTING, DEBIT, CREDIT

Вступ

Робота з великим обсягом даних завжди складна для обробки. Процеси обробки та передачі накопичуються, перетинаються, що за неправильної організації веде до втрати якоїсь частини даних або помилкового результату, система може також просто перевантажитись. Тому вкрай важливо мати інструментарій для формального опису таких даних, побудови відповідної математичної моделі. Мова алгебри скінченних предикатів дає змогу не лише формально описати процес обробки даних з урахуванням деталізованого системного аналізу предметної області, але й дозволяє побудувати економну модель у вигляді логічної мережі, за рахунок декомпозиції вхідного багатомісцевого відношення у вигляді композиції бінарних [1, 2]. Відношення описуються предикатами алгебри скінченних предикатів, а введений предикат упізнавання предмету дозволяє описувати дані будь-якої природи [3, 4].

Бухгалтерський облік банку є складовою інформаційної системи банку та включає облік операцій за відповідними рахунками на підставі автоматизованих та ручних проводок, складання агрегованих та детальних звітів.

План рахунків бухгалтерського обліку банків України – це систематизований перелік рахунків бухгалтерського обліку, який використовується для детальної та повної реєстрації всіх банківських операцій з метою забезпечення потреб у складанні фінансової звітності [5].

1. Постановка задачі

Далі формально опишемо мовою алгебри предикатів процес кореспонденції балансових рахунків з урахування депозитів суб'єктів господарської діяльності та фізичних осіб, що постійно використовується в бухгалтерії.

Метод банківського обліку складається з окремих специфічних методичних прийомів, одним із 3399 яких є подвійний запис – спосіб реєстрації інформації в обліковій системі банку, згідно з яким кожен запис-реєстрація відображається за двома рахунками: за дебетом одного рахунку та одночасно за кредитом іншого[6].

Усі номери рахунків аналітичного обліку (крім рахунків бюджету) у банку формуються за такою схемою:

AAAA. B. EEEEEEEEE

1. AAAA – номер балансового рахунку (4 знаки).

2. B – ключовий розряд (1 знак).

3. EEEEEEEEE – інформація про аналітичний рахунок (до 9 знаків).

Процес кореспонденції рахунків обліку депозитів суб'єктів господарської діяльності та фізичних осіб наведено в табл. 1, яку складено за рахунками конкретного клієнта – фізичної особи визначеного банку.

Таблиця 1

Процес кореспонденції рахунків обліку депозитів суб'єктів господарської діяльності та фізичних осіб

Зміст операції, x_2	Кореспонденція рахунків, x_1	
	Дебіт	Кредіт
Розміщення депозита	**1001***3*183456 **2620***1*98305393	**2630***2*98305493 **2635***6*98305493
Погашення депозиту	**2630***2*98305493 **2635***6*98305493	**1001***3*183456 **2620***1*98305393
Нарахування відсотків	**7041***4*453	**2638***3*453
Виплата відсотків	263**638***3*453	**1001**3183456 **2620***1*98305393
Виплата відсотків авансом	**3500***2*453	**2630***2*98305493 **2635***6*98305493
Амортизація виплачених авансом відсотків з віднесенням суми на витрати	**7041**4453	**3500***2*453

Формальний опис процесу кореспонденції рахунків обліку депозитів суб'єктів господарської діяльності та фізичних осіб виконується згідно з «Планом рахунків бухгалтерського обліку комерційних банків України» [7].

Зауважимо, що у клітинках табл. 1 для фізичних осіб в Дебеті, якщо розміщено депозит, та в Кредиті під час погашення депозиту та виплати відсотків, міститься по два рахунки в залежності від того, задіяно готівку (балансовий рахунок 1001) чи кошти на поточному рахунку (балансовий рахунок 2620).

Наведено табл. 2 та 3 балансових рахунків обліку депозитів для різних типів суб'єктів господарської діяльності, а саме для фізичних осіб та юридичних осіб.

Таблиця 2

Балансовий рахунок обліку депозитів для фізичних осіб

Фізична особа		
Зміст операції	Дебіт	Кредіт
Розміщення депозита	1001 2620	2630 2635
Погашення депозиту	2630 2635	1001 2620
Нарахування відсотків	7041	2638
Виплата відсотків	2638	1001 2620
Виплата відсотків авансом	3500	2630 2635
Амортизація виплачених авансом відсотків з віднесенням суми на витрати	7041	3500

Таблиця 3

Балансовий рахунок обліку депозитів для юридичних осіб

Юридична особа		
Зміст операції	Дебіт	Кредіт
Розміщення депозита	2600	2610 2615
Погашення депозиту	2610 2615	2600
Нарахування відсотків	7021	2618
Виплата відсотків	2618	2600
Виплата відсотків авансом	3500	2610 2615
Амортизація виплачених авансом відсотків з віднесенням суми на витрати	7021	3500

2. Побудова математичної моделі процесу

Для формального опису процесу корреспонденції рахунків обліку депозитів суб'єктів господарської діяльності та фізичних осіб введемо необхідні предметні змінні: x_1 – кореспонденція рахунків зі значеннями Д (Дебіт), К (Кредіт), $x_1 \in \{\text{Д},\text{К}\}$; x_2 – зміст операції (РД – розміщення депозиту, ПД – погашення депозиту, НВ – нарахування відсотків, ВВ – виплата відсотків, ВВА – виплата відсотків авансом, А – амортизація виплачених авансом відсотків з віднесенням суми на витрати), $x_2 \in \{\text{РД},\text{ПД},\text{НВ},\text{ВВ},\text{ВВА},\text{А}\}$; x_3 – ознака того, що кошти у вигляді готівки (Г), чи знаходяться на поточному рахунку (Р), $x_3 \in \{\text{Г},\text{Р}\}$; x_4 – в залежності від строку розміщення депозиту (КС – короткостроковий, ДС – довгостроковий), $x_4 \in \{\text{КС},\text{ДС}\}$; s – тип суб'єкту господарської діяльності (фізична особа – Фіз, юридична особа – Юр), $s \in \{\text{Фіз},\text{Юр}\}$; z – балансові рахунки із визначеними значеннями,

$$z \in \{1001, 2600, 2610, 2615, 2618, 2620, 2630, 2635, 2638, 3500, 7021, 7041\}.$$

Далі необхідно пронумерувати усі клітинки парадигматичної табл. 4 .

Таблиця 4

Парадигматична таблиця процесу кореспонденції операції з нумерацією відносно її змісту

$x_2 \setminus x_1$	Д	К
РД	$\frac{1}{2}$	$\frac{9}{10}$
ПД	$\frac{3}{4}$	$\frac{11}{12}$
НВ	5	13
ВВ	6	$\frac{14}{15}$
ВВА	7	$\frac{16}{17}$
А	8	18

Для побудови адекватної математичної моделі необхідно далі виразити номери q клітинок парадигматичної табл. 4 через ознаки $x_1 - x_4$. Запишемо наступні формули:

$$x_1^{\text{Д}} x_2^{\text{РД}} x_3^{\text{Г}} = q^1;$$
$$x_1^{\text{Д}} x_2^{\text{РД}} x_3^{\text{Р}} = q^2;$$
$$x_1^{\text{Д}} x_2^{\text{ПД}} x_4^{\text{КС}} = q^3;$$
$$x_1^{\text{Д}} x_2^{\text{ПД}} x_4^{\text{ДС}} = q^4;$$
$$x_1^{\text{Д}} x_2^{\text{НВ}} = q^5;$$
$$x_1^{\text{Д}} x_2^{\text{ВВ}} = q^6;$$
$$x_1^{\text{Д}} x_2^{\text{ВВА}} = q^7;$$
$$x_1^{\text{Д}} x_2^{\text{А}} = q^8;$$
$$x_1^{\text{К}} x_2^{\text{РД}} x_4^{\text{КС}} = q^9;$$
$$x_1^{\text{К}} x_2^{\text{РД}} x_4^{\text{ДС}} = q^{10};$$
$$x_1^{\text{К}} x_2^{\text{ПД}} x_3^{\text{Г}} = q^{11};$$
$$x_1^{\text{К}} x_2^{\text{ПД}} x_3^{\text{Р}} = q^{12};$$
$$x_1^{\text{К}} x_2^{\text{НВ}} = q^{13};$$
$$x_1^{\text{К}} x_2^{\text{ВВ}} x_3^{\text{Г}} = q^{14};$$
$$x_1^{\text{К}} x_2^{\text{ВВ}} x_3^{\text{Р}} = q^{15};$$
$$x_1^{\text{К}} x_2^{\text{ВВА}} x_4^{\text{КС}} = q^{16};$$
$$x_1^{\text{К}} x_2^{\text{ВВА}} x_4^{\text{ДС}} = q^{17};$$
$$x_1^{\text{К}} x_2^{\text{А}} = q^{18}.$$

Таким чином, було отримано систему предикатних рівнянь, що повністю описують парадигматичну табл. 4.

2.1. Формальний опис значень кореспонденції, змісту операції, виду коштів та терміну розміщення депозиту

Для побудови логічної мережі, що відображає процес корреспонденції рахунків обліку депозитів суб'єктів господарської діяльності та фізичних осіб необхідно спочатку застосувати операцію почленної диз'юнкції до усіх отриманих рівнянь, далі переозначити отримані рівняння і знову об'єднати і так до тих пір, поки не залишаться лише бінарні відношення.

З метою отримання економної множини імен впливів, що визначають процес кореспонденції у бухгалтерському обліку (значень кореспонденції, змісту операції, виду коштів та терміну розміщення депозиту), виконуємо операцію почленної диз'юнкції якомога більшого числа споріднених рівностей:

$$x_1^{\text{Д}} x_2^{\text{РД}} x_3^{\text{Г}} = q^1;$$
$$x_1^{\text{Д}} x_2^{\text{РД}} x_3^{\text{Р}} = q^2;$$
$$x_1^{\text{Д}} x_2^{\text{ПД}} x_4^{\text{КС}} = q^3;$$
$$x_1^{\text{Д}} x_2^{\text{ПД}} x_4^{\text{ДС}} = q^4;$$
$$x_1^{\text{Д}} x_2^{\text{НВ}} = q^5;$$
$$x_1^{\text{Д}} x_2^{\text{ВВ}} = q^6;$$
$$x_1^{\text{Д}} x_2^{\text{ВВА}} = q^7;$$
$$x_1^{\text{Д}} x_2^{\text{А}} = q^8;$$
$$x_1^{\text{К}} (x_2^{\text{РД}} \vee x_2^{\text{ВВА}}) x_4^{\text{КС}} = q^9 \vee q^{16};$$
$$x_1^{\text{К}} (x_2^{\text{РД}} \vee x_2^{\text{ВВА}}) x_4^{\text{ДС}} = q^{10} \vee q^{17};$$
$$x_1^{\text{К}} (x_2^{\text{ПД}} \vee x_2^{\text{ВВ}}) x_3^{\text{Г}} = q^{11} \vee q^{14};$$
$$x_1^{\text{К}} (x_2^{\text{ПД}} \vee x_2^{\text{ВВ}}) x_3^{\text{Р}} = q^{12} \vee q^{15};$$
$$x_1^{\text{К}} x_2^{\text{НВ}} = q^{13};$$
$$x_1^{\text{К}} x_2^{\text{А}} = q^{18};$$

Формуємо функцію переходу від номерів клітинок парадигматичної таблиці q до номерів впливів, що визначають процес кореспонденції у бухгалтерському обліку, r у вигляді наступних предикатних рівнянь:

$$q^1 = r^1;$$
$$q^2 = r^2;$$
$$q^3 = r^3;$$
$$q^4 = r^4;$$
$$q^5 = r^5;$$
$$q^6 = r^6;$$
$$q^7 = r^7;$$
$$q^8 = r^8;$$
$$q^9 \vee q^{16} = r^9;$$
$$q^{10} \vee q^{17} = r^{10};$$
$$q^{11} \vee q^{14} = r^{11};$$
$$q^{12} \vee q^{15} = r^{12};$$
$$q^{13} = r^{13};$$
$$q^{18} = r^{14}.$$

Вводимо номер впливів, що визначають процес кореспонденції у бухгалтерському обліку, перенумеровуючи клітинки парадигматичної табл. 4, отримаємо табл. 5.

Таблиця 5

Парадигматична таблиця з нумерацією впливів, що визначають процес кореспонденції

$x_2 \setminus x_1$	Д	К
РД	1 2	9 10
ПД	3 4	11 12
НВ	5	13
ВВ	6	11 12
ВВА	7	9 10
А	8	14

Формуємо залежність номеру впливів, що визначають процес кореспонденції у бухгалтерському обліку, r від змінних x_1, x_2, x_3, x_4:

$$x_1^{\text{Д}} x_2^{\text{РД}} x_3^{\text{Г}} = r^1;$$
$$x_1^{\text{Д}} x_2^{\text{РД}} x_3^{\text{Р}} = r^2;$$
$$x_1^{\text{Д}} x_2^{\text{ПД}} x_4^{\text{КС}} = r^3;$$
$$x_1^{\text{Д}} x_2^{\text{ПД}} x_4^{\text{ДС}} = r^4;$$
$$x_1^{\text{Д}} x_2^{\text{НВ}} = r^5;$$
$$x_1^{\text{Д}} x_2^{\text{ВВ}} = r^6;$$
$$x_1^{\text{Д}} x_2^{\text{ВВА}} = r^7;$$
$$x_1^{\text{Д}} x_2^{\text{А}} = r^8;$$
$$x_1^{\text{К}} (x_2^{\text{РД}} \vee x_2^{\text{ВВА}}) x_4^{\text{КС}} = r^9;$$
$$x_1^{\text{К}} (x_2^{\text{РД}} \vee x_2^{\text{ВВА}}) x_4^{\text{ДС}} = r^{10};$$
$$x_1^{\text{К}} (x_2^{\text{ПД}} \vee x_2^{\text{ВВ}}) x_3^{\text{Г}} = r^{11};$$
$$x_1^{\text{К}} (x_2^{\text{ПД}} \vee x_2^{\text{ВВ}}) x_3^{\text{Р}} = r^{12};$$
$$x_1^{\text{К}} x_2^{\text{НВ}} = r^{13};$$
$$x_1^{\text{К}} x_2^{\text{А}} = r^{14};$$

Проводимо бінаризацію записаного відношення, зв'язуючи змінну r зі змінними x_1, x_2, x_3, x_4: Для цього знаходимо відношення P_1, яке зв'язує змінні x_1, та r:

$$P_1(x_1, r) = x_1^{\text{Д}} \left(r^1 \vee r^2 \vee r^3 \vee r^4 \vee r^5 \vee r^6 \vee r^7 \vee r^8 \right) \vee$$
$$\vee x_1^{\text{К}} \left(r^9 \vee r^{10} \vee r^{11} \vee r^{12} \vee r^{13} \vee r^{14} \right)$$

Отриманні відношення зображуються у вигляді дводольних графів:

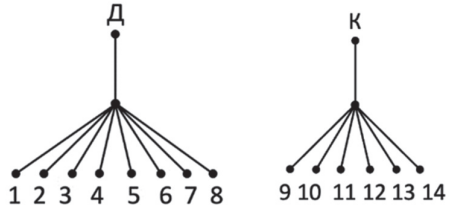


Рис. 1. Графи залежності значень кореспонденції рахунків x_1 від інших впливів, що визначають процес кореспонденції у бухгалтерському обліку, r

Будуємо відношення P_2, яке зв'язує змінні x_2 та r:

$$P_2(x_2,r) = x_2^{\text{РД}}\left(r^1 \vee r^2\right) \vee (x_2^{\text{РД}} \vee x_2^{\text{ВВА}})$$
$$\left(r^9 \vee r^{10}\right) \vee \vee x_2^{\text{ПД}}\left(r^3 \vee r^{14}\right) \vee (x_2^{\text{ПД}} \vee x_2^{\text{ВВ}}) \wedge$$
$$\wedge\left(r^{11} \vee r^{12}\right) \vee x_2^{\text{НВ}}\left(r^5 \vee r^{13}\right) \vee x_2^{\text{ВВ}} r^6 \vee x_2^{\text{ВВА}} r^7 \vee$$
$$\vee x_2^{\text{А}}\left(r^8 \vee r^{14}\right).$$

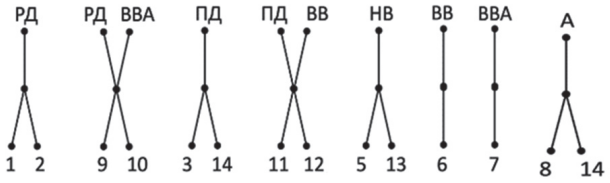


Рис. 2. Графи залежності змісту операції x_2 від інших впливів, що визначають процес кореспонденції у бухгалтерському обліку, r

Будуємо відношення P_3, яке зв'язує змінні x_3 та r:

$$P_3(x_3,r) = x_3^{\text{г}}\left(r^1 \vee r^{11}\right) \vee x_3^{\text{р}}\left(r^2 \vee r^{12}\right) \vee (x_3^{\text{г}} \vee x_3^{\text{р}})(r^3 \vee$$
$$\vee r^4 \vee r^5 \vee r^6 \vee r^7 \vee r^8 \vee r^9 \vee r^{10} \vee r^{13} \vee r^{14}).$$

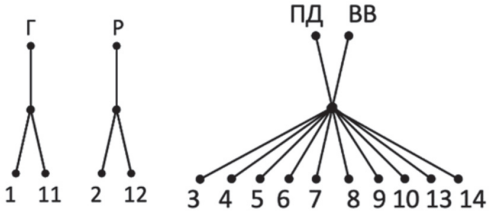


Рис. 3. Графи залежності виду коштів x_3 від інших впливів, що визначають процес кореспонденції у бухгалтерському обліку, r

Будуємо відношення P_4, яке зв'язує змінні x_4 та r:

$$P_4(x_4,r) = x_4^{\text{КС}}\left(r^3 \vee r^9\right) \vee x_4^{\text{ДС}}\left(r^4 \vee r^{10}\right) \vee (x_4^{\text{КС}} \vee x_4^{\text{ДС}}) \wedge$$
$$\wedge\left(r^1 \vee r^2 \vee r^5 \vee r^6 \vee r^7 \vee r^8 \vee r^{11} \vee r^{12} \vee r^{13} \vee r^{14}\right).$$

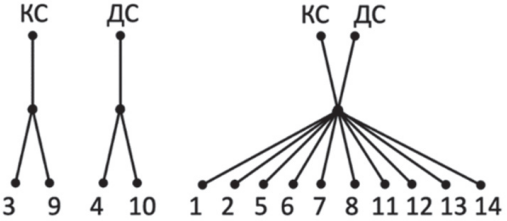


Рис. 4. Графи залежності терміну розміщення депозиту x_4 від інших впливів, що визначають процес кореспонденції у бухгалтерському обліку, r

2.2. Формальний опис структури балансового рахунку в залежності від впливів, що визначають процес кореспонденції у бухгалтерському обліку

Розглянемо структуру балансового рахунку. Перші два знаки у балансовому рахунку називають розділом балансового рахунку ($z_{\text{р}}$). Розділ балансового рахунку визначається виключно впливами, що визначають процес кореспонденції у бухгалтерському обліку (ознаки $x_1 - x_4$), тоді як на 3-й знак балансового рахунку (z_3) впливає також тип суб'єкту господарювання, а саме процес кореспонденції рахунків з урахування депозитів проводиться для фізичної особі, чи для юридичної особи (s).

Побудуємо відношення P_5, яке зв'язує номер впливів, що визначають процес кореспонденції у бухгалтерському обліку r, з 3-м знаком у балансовому рахунку. Цей зв'язок відображено у таблиці:

Таблиця 6

Зв'язок номерів впливів, що визначають процес кореспонденції у бухгалтерському обліку r, з 3-м знаком у балансовому рахунку

$x_2 \setminus x_1$	Д	К
РД	2,0 0,-	3,1 3,1
ПД	3,1 3,1	2,0 0,-
НВ	4,2	3,1
ВВ	3,1	2,0 0,-
ВВА	0,0	3,1 3,1
А	4,2	0,0

Запишемо відношення P_5 у вигляді формули:

$$P_5(z_3,r) = (z_3^2 \vee z_3^0)\left(r^1 \vee r^{11}\right) \vee z_3^0\left(r^2 \vee r^{12}\right) \vee (z_3^3 \vee$$
$$\vee z_3^1)\left(r^3 \vee r^4 \vee r^6 \vee r^9 \vee r^{10} \vee r^{13}\right) \vee (z_3^4 \vee z_3^2)(r^5 \vee$$
$$\vee r^8) \vee (z_3^0 \vee z_3^0)\left(r^7 \vee r^{14}\right).$$

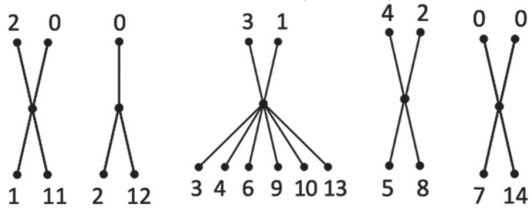


Рис. 5. Графи залежності 3-ого знака у балансовому рахунку z_3 від впливів, що визначають процес кореспонденції у бухгалтерському обліку, r

Відношення P_6, яке зв'язує номер впливів, що визначають процес кореспонденції у бухгалтерському обліку r, з розділом балансового рахунку $z_{\text{р}}$, представлено у табл. 7:

Таблиця 7

Зв'язок номерів впливів, що визначають процес кореспонденції у бухгалтерському обліку r, з розділом балансового рахунку $z_{\text{р}}$

$x_2 \setminus x_1$	Д	К
РД	26 10	26 26
ПД	26 26	26 10
НВ	70	26
ВВ	26	26 10
ВВА	35	26 26
А	70	35

Будуємо відношення P_6 у вигляді формули:

$$P_6\left(z_\text{р},r\right)=z_\text{р}^{26}\left(r^1\vee r^3\vee r^4\vee r^6\vee r^9\vee r^{10}\vee r^{11}\vee r^{13}\right)\vee$$
$$\vee z_\text{р}^{10}\left(r^2\vee r^{12}\right)\vee z_\text{р}^{70}\left(r^5\vee r^8\right)\vee z_\text{р}^{35}\left(r^7\vee r^{14}\right)$$

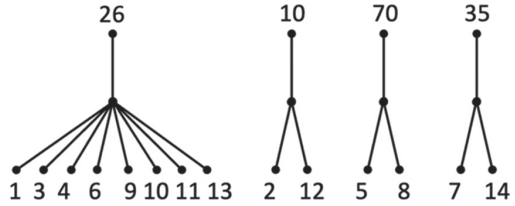


Рис. 6. Графи залежності розділу балансового рахунку $z_\text{р}$ від впливів, що визначають процес кореспонденції у бухгалтерському обліку, r

Відношення P_7, та P_8, зв'язують балансовий рахунок z з його розділом $z_\text{р}$ та 3-м знаком у структурі z_3.

$$P_7\left(z_3,z\right)=z_3^0\left(z^{1001}\vee z^{2600}\vee z^{3500}\right)\vee$$
$$\vee z_3^1\left(z^{2610}\vee z^{2615}\vee\vee z^{2618}\right)\vee z_3^2\left(z^{2620}\vee z^{7021}\right)\vee$$
$$\vee z_3^3(z^{2630}\vee z^{2635}\vee$$
$$\vee z^{2638})\vee z_3^4 z^{7041}.$$

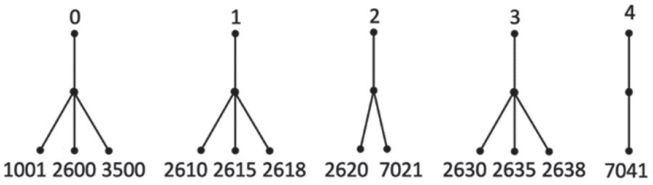


Рис. 7. Графи залежності 3-ого знака у балансовому рахунку z_3 від балансового рахунку z

$$P_8\left(z_\text{р},z\right)=z_\text{р}^{10}z^{1001}\vee$$
$$\vee z_\text{р}^{26}\left(z^{2600}\vee z^{2610}\vee z^{2615}\vee z^{2618}\vee z^{2620}\vee z^{2630}\vee z^{2635}\vee z^{2638}\right)\vee$$
$$\vee z_\text{р}^{35}z^{3500}\vee z_\text{р}^{70}\left(z^{7021}\vee z^{7041}\right)$$

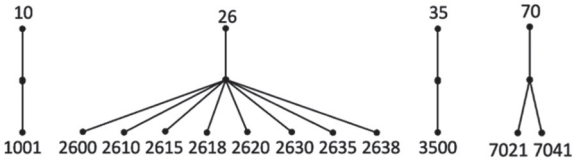


Рис. 8. Графи залежності розділу балансового рахунку $z_\text{р}$ від балансового рахунку z

Відзначимо, що 3-й знак в балансовому рахунку (z_3) пов'язано з типом суб'єкту господарювання (s), а саме процес кореспонденції рахунків обліку депозитів проводиться для фізичної особі чи для юридичної особи. Тип суб'єкту господарювання s пов'язано з 3-м знаком балансового рахунку z відношенням P_9, яке наводиться у табл. 8:

Таблиця 8

Зв'язок типу суб'єкту господарювання s з 3-м знаком балансового рахунку z

s	Фіз	Юр
z_3	0,2,3,4	0,1,2

$$P_9\left(s,z_3\right)=s^{\text{Фіз}}\left(z_3^0\vee z_3^2\vee z_3^3\vee z_3^4\right)\vee s^{\text{Юр}}\left(z_3^0\vee z_3^1\vee z_3^2\right)$$

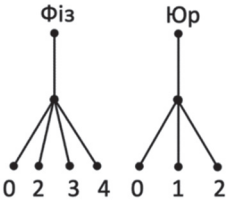


Рис. 9. Графи залежності типу суб'єкту господарювання, s від 3-ого знаку у балансовому рахунку z_3

Балансові рахунки мають поділятися за ознакою q зі значеннями синтетичного рахунку (активний (А), пасивний (П) та активно-пасивний (АП)). Складемо таблицю залежності балансового рахунку z від ознаки q, та відношення P_{10}.

Таблиця 9

Таблиця залежності балансового рахунку z від ознаки q

q	А	П	АП
z	1001 3500	2610 2615 2630 2635	2600 2620 2618 2638 7021 7041

$$P_{10}\left(q,z\right)=q^{\text{А}}\left(z^{1001}\vee z^{3500}\right)\vee$$
$$\vee q^{\text{П}}\left(z^{2610}\vee z^{2615}\vee z^{2630}\vee\vee z^{2635}\right)\vee$$
$$q^{\text{АП}}(z^{2600}\vee z^{2620}\vee z^{2618}\vee z^{2638}\vee z^{7021}\vee z^{7041})$$

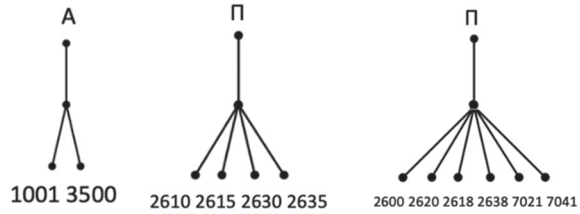


Рис. 10. Графи залежності ознаки синтетичного рахунку q від балансового рахунку z

Розділ балансового рахунку $z_\text{р}$ можемо розглянути с точки зору приналежності до типу рахунку y, а саме клієнтський (КЛ) чи банківський (Б). Складемо таблицю залежності розділу балансового рахунку $z_\text{р}$ від ознаки y, у вигляді відношення P_{11} (таблиця 10).

Таблиця 10

Таблиця залежності розділу балансового рахунку $z_\text{р}$ від ознаки y

y	КЛ	Б
$z_\text{р}$	10 26	35 70

$$P_{11}\left(z_\text{р},y\right)=y^{\text{КЛ}}\left(z_\text{р}^{10}\vee z_\text{р}^{26}\right)\vee y^{\text{Б}}\left(z_\text{р}^{35}\vee z_\text{р}^{70}\right).$$

Рис. 11. Графи залежності типу рахунку y від розділу балансового рахунку $z_\text{р}$

3. Побудова логічної мережі процесу кореспонденції рахунків бухгалтерського обліку

Математична модель процесу кореспонденції рахунків обліку депозитів суб'єктів господарської діяльності та фізичних осіб представляє собою багатомісцевий предикат, що залежить від 11 змінних. Цей предикат характеризується системою бінарних відношень P_1 - P_{10}, що задано дводольними графами та формулами відповідних предикатів. Предикат моделі має вигляд кон'юнкції усіх побудованих вище бінарних предикатів:

$$P(x_1,x_2,x_3,x_4,r,s,z_p,z_3,z,q,y) =$$
$$= P_1(x_1,r)P_2(x_2,r)P_3(x_3,r)P_4(x_4,r)P_5(z_3,r)P_6(z_p,r)$$
$$P_7(z_3,z)P_8(z_p,z)P_9(s,z_3)P_{10}(q,z)P_{11}(z_p,y)$$

Предикату моделі відповідає відношення моделі P, що зв'язує між собою предметні змінні $x_1,x_2,x_3,x_4,r,s,z_p,z_3,z,q,y$. Відношення моделі P наглядно зображуємо у вигляді логічної мережі.

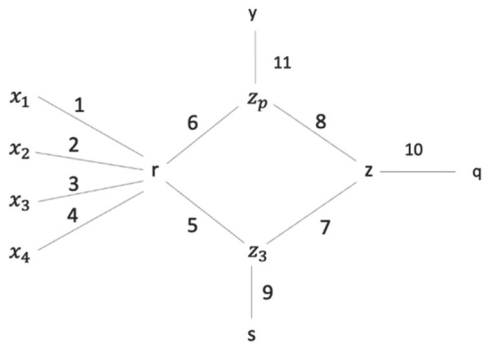


Рис. 12. Логічна мережа

Висновки

Побудована логічна мережа складається з вузлів та гілок. У кожному вузлі знаходиться предметна змінна, яку ще називають атрибутом цього вузла. З кожним вузлом зв'язується його домен, тобто область значень атрибута. Це знання називається станом вузла. Область визначення усіх змінних було визначено в постановці задачі. Мережа працює ітераційно. На першому кроці задаються якісь знання на один чи декілька вузлів, на наступному кожен вузол мережі змінює свій стан, і так до тих пір, поки не отримаємо сталі стани мережі на двох кроках підряд.

Побудована логічна мережа дозволяє виконувати задачі аналізу та синтезу. Інструментарій алгебри предикатів за рахунок введення предикату упізнавання предметів, розбиття саме на бінарні предикати та розпаралелювання обробки різних станів одночасно забезпечує швидку та якісну обробку інформації [8] .

Список літератури:

[1] *Bondarenko M.F.,* Shabanov-Kushnarenko Yu.P. Mozghopodobnie strukturi: spravochnoe posobye. Tom pervii. – K.: Naukova dumka, 2011. – 460 s.

[2] *Vechirska I.D.,* Vechirska A.D. Analiz zastosuvannia instrumentariiu alhebry predykativ dlia otsinennia yakosti skladnoi systemy //II International Scientific and Practical Conference «Ricerche Scientifiche E Metodi Della Loro Realizzazione: Esperienza Mondiale E Realtà Domestiche», 26.11.2021 . Vena, AUT/ Band2. – P.48-50.

[3] *Vechirska I.D.* Analiz metodu pobudovy ta pryntsypiv roboty reliatsiinoi merezhi yak bahatorivnevoi struktury paralelnoi dii // Bionika intelektu: nauk.-tekhn. zhurnal. – 2013. – № 2 (81). – S.15 – 21.

[4] *Gorokhovatskyi, V.A.,* Vechirska, I.D. & Chetverikov, G.G. (2016). Method for building of logical data transform in the problem of establishing links between the objects in intellectual telecommunication systems. // Telecommunications and engineering, (18, Vol. 75), P. 1645-1655.

[5] Natsionalni standarty bukhhalterskoho obliku v Ukraïnii : navchalnyy posibnyk. Normatyvno-praktychnyy material. Stanom na 1 lystopada 2000 r. / za red. d.e.n., dots. R. L. Khomiaka. – Lviv : Intelekt-Zakhid, 2000. – 444 s. – ISBN 966-7597-03-2.

[6] *Kindratska L.M.* Bukhhalterskyy oblik u komertsiy nykh bankakh Ukraïnyi : pidruchnyk / L. M. Kindratska. – K. : KNEU, 2000.

[7] Postanova Pravlinnia NBU «Instruktsii pro zastosuvannia Planu rakhunkiv bukhhalterskoho obliku bankiv Ukraïny» : vid 11.09.2017 roku N 89 // Zakonodavchi ta normatyvni akty z bankivskoï diialnosti. – 2017.

[8] *Vechirska, I.D.,* Honcharov, I.E. & Shepilov, S.I. Doslidzhennia lohiky skinchennykh predykativ yak kompozytsiinonominatyvnoi lohiky // Bionika intelektu: nauk.-tekhn. zhurnal. – 2014. – № 2 (83). – S. 53-60.

Надійшла до редколегії 16.10.2021

UDC 004.8

DOi 10.30837/ bi.2021.2(97).03

Yana Daniiel[1], Kostiantyn Onyshchenko[2], N. Kameniuk[3]

[1]Assistant of the Department of Software engineering,
Kharkiv National university of Radio electronics, Kharkiv, Ukraine,
yana.daniiel@nure.ua, ORCID ID: 0000-0002-3895-0744

[2]Assistant of the Department of Software engineering,
Kharkiv National university of Radio electronics, Kharkiv, Ukraine,
kostiantyn.onyshchenko@nure.ua, ORCID ID: 0000-0002-7746-4570

[3]Master's student of Riga Technical University, Latvia, nataliia.kameniuk@gmail.com

USAGE OF LSTM MODELS FOR NATURAL LANGUAGE UNDERSTANDING

The problem of emotion classification is a complex task of language interpretation. In this work, a number of existing solutions for emotional classification problem were considered. The evaluation of performance of the considered models was conducted. The model for emotion classification in three-sentence conversations is proposed in this work. The model is based on smileys and word embeddings with domain specificity in state of art conversations on the Internet. The model performance is evaluated and compared with language processing model BERT. The proposed model is better at classifying emotions than BERT (F1 78 versus 75). However, modern performance of models for language representation did not achieve the human performance due to the complexity of natural language. There is a variety of factors to consider when choosing the word embeddings and training methods to design the model architecture.

NATURAL LANGUAGE PROCESSING, NEURAL NETWORK, NATURAL LANGUAGE

Данієль Я.Д., Онищенко К.Г., Каменюк Н. Використання LSTM-моделей для обробки природної мови. Проблема класифікації емоцій є однією із важливих завдань інтерпретації мови. У даній роботі було розглянуто ряд існуючих рішень проблеми емоційної класифікації. Проведено оцінку продуктивності розглянутих моделей. Запропоновано модель класифікації емоцій у розмовах із трьох речень. Модель заснована на смайлах і використанням слів із оглядом на специфіку сучасного спілкування в в Інтернеті. Продуктивність моделі оцінюється та порівнюється з моделлю обробки мови BERT. Запропонована модель краще класифікує емоції, ніж BERT (F1 78 проти 75). Однак, сучасне виконання моделей мовного представлення не досягло продуктивності людини через складність природної мови. Існує ряд факторів, які слід враховувати при виборі вбудовування слів і методів навчання для проектування архітектури моделі.

ОБРОБКА ПРИРОДНОЇ МОВИ, НЕЙРОННА МЕРЕЖА, ПРИРОДНА МОВА

Даниель Я.Д., Онищенко К.Г., Каменюк Н. Использование LSTM-моделей для обработки есстественного языка. Проблема классификации эмоций является одной из важнейших задач интерпретации языка. В данной работе был рассмотрен ряд существующих решений проблемы эмоциональной классификации. Проведена оценка производительности рассматриваемых моделей. Предложена модель классификации эмоций в разговорах из трех предложений. Модель основана на смайлах и использованием слов с учетом специфики современного общения в Интернете. Производительность модели оценивается и сравнивается с моделью обработки языка BERT. Предлагаемая модель лучше классифицирует эмоции, чем BERT (F1 78 против 75). Однако, современное исполнение моделей речевого представления не достигло производительности человека из-за сложности естественной речи. Есть ряд факторов, которые следует учитывать при выборе встраивания слов и методов обучения для проектирования архитектуры модели.

ОБРАБОТКА ПРИРОДНОГО ЯЗЫКА, НЕЙРОННАЯ СЕТЬ, ПРИРОДНЫЙ ЯЗЫК

Introduction

Over the past few decades, the amount of text data produced by humanity has grown exceedingly. One of the reasons behind this fact is that we actively exchange information and publish our thoughts on websites and social media.

Such unstructured data is represented in arbitrary form and is often complemented by emojis, which makes it difficult for a computer to categorize and derive meaning from the source. On this basis, the challenge to teach computers to properly process this information arose.

Natural language processing (NLP) is widely used for text data analysis and classification [11].

The core aspects of language understanding include three parameters, which are morphology, semantics, and syntax. Morphology is the study of word or statement structure; semantics is the study of meaning, reference, or truth; syntax is the study of how words and morphemes combine to form larger units such as phrases and sentences [3].

Modern models consider all three parameters. The models are using data-driven approach through machine learning and deep learning.

The goal of this work is to consider existing solutions for text data processing in terms of emotional classification and propose the model that can solve such task.

1. Problem statement

The semantic evaluation problem of emotion classification will be considered in this work. The deep learning approach will be based on Keras and TensorFlow – the

Python frameworks. These frameworks have ready to use functions for rapid optimization, model prediction, model tuning and many more. This will allow to achieve approximate state of the art results with an original model architecture. BERT will also be implemented to obtain a current state of the art model in natural language understanding. The received results will be evaluated and compared [4].

The theoretical fundamentals of emotional classification models will be discussed. The practical details of the proposed model's training and BERT on unseen test data will be presented. The obtained results will be evaluated and compared. The differences between the models will be illustrated.

It should be considered that the proposed model will be implemented based on Keras and TensorFlow. The learned models will be applied to a specific pre-defined dataset to solve the semantic evaluation problem of emotion classification. The models will be trained on four CPU cores. These conditions apply specific limitations on the scope of the project.

2. Analysis of existing solutions

1) Machine Learning

Machine learning is a modelling approach focused on finding underlying patterns in a dataset. An algorithm with learning function is applied to rich dataset. The parameters of model are modified to enhance the predictive performance of a model. The unseen data is used to evaluate the trained model.

a) 3.1.1 RNN

Recurrent Neural Network (RNN) is a class of artificial neural networks, where connections between nodes form a directed sequential graph. By this, the sequential nature of input can be considered when making output predictions [12].

A set of feedback weights contained in a hidden state vector is computed at every step in the sequence that pass information from earlier time points. The ability to predict to which class the sequence belongs to is provided by the model reformulation. This will allow to incorporate new time dependency by using the final recurrent hidden state vector to make softmax probability predictions.

$$\hat{y} = \varphi\left(Vh_p\right), \tag{1}$$

where

$$h_p = \sigma\left(Ux_p + Wh_{p-1}\right). \tag{2}$$

The parametrization W of the model is comprised of the weight matrices U, V and W. The hidden layer is dependent on earlier states.

The Fig. 1 provides more information about the recurrence mechanism applied.

New hidden state vector is computed at each time step. The vector is trained to pass forward the most significant information for solving the problem through gradient descent with an appropriate loss function [12].

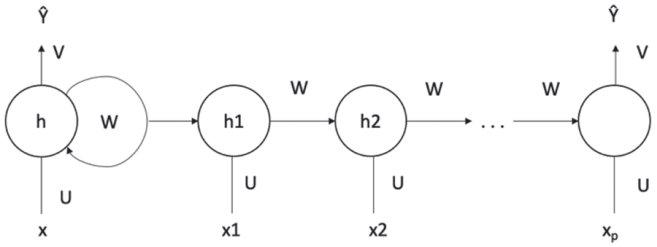


Fig. 1. The unfolding concept

b) LSTM

Long Short-Term Memory (LSTM) networks have two enhancements comparing to previously considered RNNs. The first enhancement is that each time step passes a hidden state vector and a local context vector to the next recurrent node. The second enhancement is that long short-term memory network contains a set of gating mechanisms (Fig. 2). These mechanisms provide the ability to the model to decide which data to pass forward in recurrence. These enhancements allow the LSTM model to learn long-term dependencies in the sequence more stably.

The gating mechanisms of LSTM contain an input gate, a context gate, forgetting gate and an output gate, which are activated matrix manipulations. The manipulations are based on gating weights optimally learned through training. The following relationships define the gates and their associated weights [1].

$$f_p = \sigma(x_p U^f + h_{p-1} W^f), \tag{3}$$

$$i_p = \sigma(x_p U^i + h_{p-1} W^i), \tag{4}$$

$$o_p = \sigma(x_p U^o + h_{p-1} W^o), \tag{5}$$

The functions are recurrent to hidden state of the previous time step and the current input data at this time.

A candidate $c_p^{\sim}$ for the context state $c_p^{\sim}$ is computed by

$$c_p^{\sim} = \tanh(x_p U^c + h_{p-1} W^c). \tag{6}$$

The previous context information filtered by the forgetting gate and present information from input gate in an equation filtered through the context gate form the context state.

The gating machinery provides an ability to determine which long-term and short-term data to filter and pass to the final output representation [10].

The hidden state representation of the sequence h_{p-1} is computed as a combination of the filtered output from the output gate and current context information.

$$h_p = o_p \tanh(c_p). \tag{7}$$

The received output state can be used for prediction. This output can be used in the same way as the hidden states were used in RNN architecture equation (1) considered earlier.

On this basis, LSTM can represent complex sequences and make stable predictions.

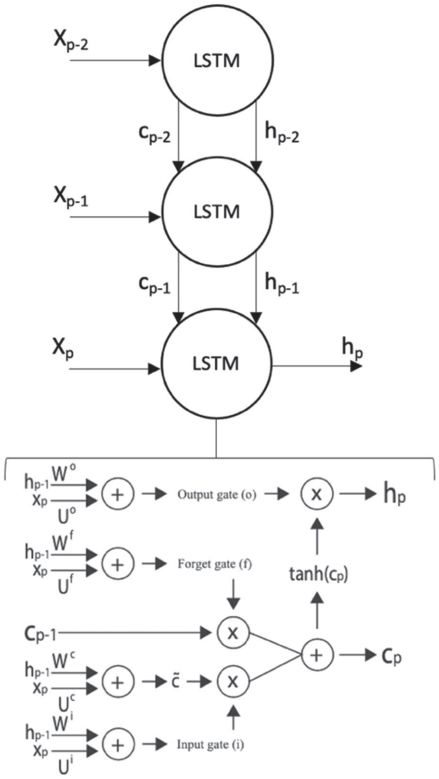


Fig. 2. The LSTM node

c) Bidirectionality

It should be considered that not all words can be predicted by the words before them.

In the sequences like “bow tie” and “bow ring”, “bow” is contextually dependent both on the word sequences before and after them. This is a challenge for regular recurrent networks. The solution for this challenge is bidirectionality, which means reversing direction of the sequence and feeding it to network. Both resulting hidden states are concatenated, which is a standard practice for many language models [10].

In Fig. 3, the simple RNN from Fig. 1 is extended to work bidirectionally.

The states h in Fig. 3 can be from regular nodes or LSTM. The network with the reverted states is a copy from the original network. The resulting hidden states are concatenated into form

$$h_p = (h^{\rightarrow}_p, h_1^{\leftarrow}) \quad (8)$$

The final hidden state vector can be used in the same way as for RNN through equation (8). The bidirectional models can be modelled by doubling the number of weights. This method is often applied to improve model representation and predictions due to its ability to add contextual information to language models [5].

2) Vector Representation of Language

Machine Learning models represented above have real valued vectors xi. The language is represented as vectors which is an essential step in using neural networks as emotional classifiers. The words w are gathered in a vocabulary to form the bag of words [2].

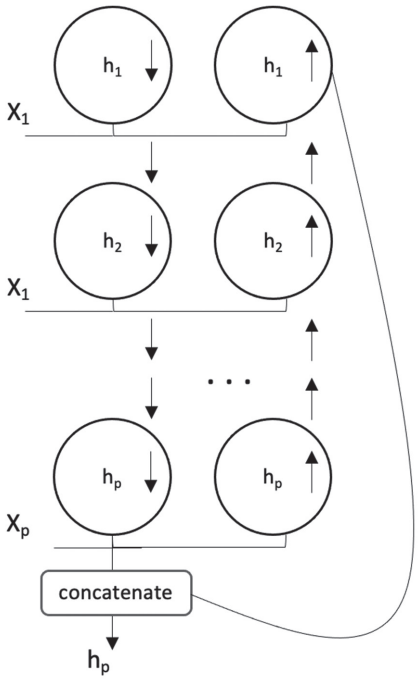


Fig. 3. Bi-directional RNN

$$V = \{w_i : i \in 1, ..., N\} \quad (8)$$

The phrase is divided into one-word hot vectors, which do not provide any extra information about the context. The vectors are also computationally hard to use as a single vector is a single word.

a) Word Embeddings

A more efficient solution is to pass a lower dimensional vector $x_i \in Rd$ ($d<N$) which also contains language information about the word wi. Less computational power is required to process such vector by a classificational model. This vector contains more language information than one-word hot vector. This type of lower dimensional representation is called word embedding.

Statistical language modelling is the other language feature. The words are used in conjunction with each other. This can get a lot of information about the semantic and syntactical use of a word through the context [9].

Principal Component Analysis (PCA) is a classic dimensional reduction technique, which is based on singular value decomposition of the co-occurrence matrix. This approach is does not need linguistic rules to be fed to the system, which makes it unsupervised. It considers the text corpus in a whole, in different context, which also makes it computationally expensive.

This can be solved by using a feed forward neural network to predict the n-gram probabilities of the words in the vocabulary given the context that came before [9].

Trainable d-dimensional random initialized vector represents each word and n previous words are used as a context. These vectors are concatenated and fed through several hidden layers. The output layer is a softmax probability over the vocabulary of all the probabilities to meet each word by the context words. The network is learning

linguistically valuable information in the matrix of d-dimensional vectors while training to predict n-grams.

On this basis, words embedding can be used as the sole input to other machine learning models for real NLP tasks, such as machine translation, and semantic sentence parsing. The Fig.4 illustrates a scheme of such feed forward neural network.

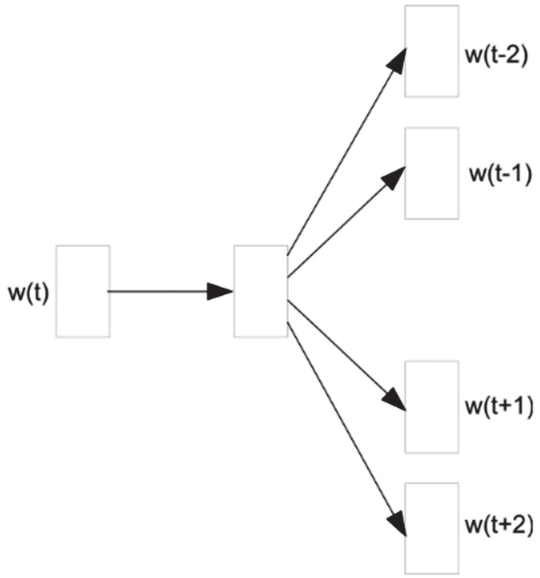


Fig. 4. Bi-directional RNN

It should be considered that this model is expensive to train. To increase the context window for the network to learn, there is a need to increase the number of input nodes, and the task of emotional classification requires a lot of context.

3) GloVe

The skip-gram model is another proposed RNN approach which is efficient on smaller datasets with better context information. RNNs efficiently model rich context information due to the fact they have a sequence memory of the previously met words. The model starts with the opposite probability comparing to the n-gram prediction, because this is the probability of different context words. The basis of this probability is a target word [6].

GloVe is the other unsupervised learning algorithm for obtaining vector representations for words. The difference of GloVe from the skip-gram and feed forward word embedding is the fact that GloVe is trained on a loss function. The loss function considers both local co-occurrences from the n-length context windows and global count-based co-occurrence probabilities from the text corpus. This allows to encode more of the language features comparing to PCA. This ability is provided since including only local context information does not give enough information to features about the frequency the words occur in rare contexts [7]. The loss function has slightly richer context information embedded in the vectors comparing to skip-gram model.

4) BERT

Pre-trained word embeddings in language are easy to use and flexible. Their drawback is that one vector can represent only one word. This means homonyms or phrasal verbs, like ‘key’ or ‘get’ lose their language information. This can be mitigated by using RNN or LSTM to carry information over sequences to create a context for a single word, but these networks cannot provide rich context data to predict context-heavy language constructions.

One of the proposed approaches to solve this issue is BERT (Bi-Directional Representations from Transformers). In its core, the transformer relies on attention for its representation to improve the contextual awareness [2].

The weighted representation of all hidden state vectors is trained at the same time with the recurrence relationships. This allows the network to be aware of the previous hidden states and do not rely solely on the final hidden state representation when making predictions.

The transformers apply attention to underlying word vectors from the original sequence to extract relevant language features. This is called text encoding. Since transformers do not use any recurrent relations, this allows to conduct training in parallel [2].

On this basis, BERT model is applicable to solve NLP tasks. BERT pre-trained weights can be downloaded via the Internet and used for transfer learning. This will allow to compare BERT with the other models in the task of emotion classification.

a) Structure of the model

BERT is a sequence-to-sequence language representation model, which is efficient for NLP tasks solution. A sequence of language information X = (I0, . . . , In) as inputs and outputs a contextualized vector representation H = (h0, . . . , hn) of the elements of the input sequence.

During the pre-processing phase, the model splits words into word-parts. The placeholder tags are inserted before the sequence and after sentences, however this does not allow input vectors to directly correspond to the underlying words in the sequence.

Due to the design of BERT model, the output presentation h0 becomes a distributed representation of the underlying sequence. This sequence is similar to the final hidden state of RNN. A classification model can be obtained when adding an extra hidden layer to the BERT model and activating this model with a softmax function [11].

$$\hat{y} = \varphi\left(Vh_0\right) \tag{9}$$

Encoders, which are stacking nodes, are the other part of this language representation framework. Encoders are used to create encoded text representation (Fig. 5).

Every encoder layer abstracts language patterns from an input sequence. It allows to form more complicated patterns as the information flows up the layers. The first encoder layer L receives language inputs and the last

encoder layer outputs the final encoded language information [11].

$$H_1 = Encoder(X)$$
$$\ldots \quad (10)$$
$$H_L = Encoder\,(H_{L-1})$$

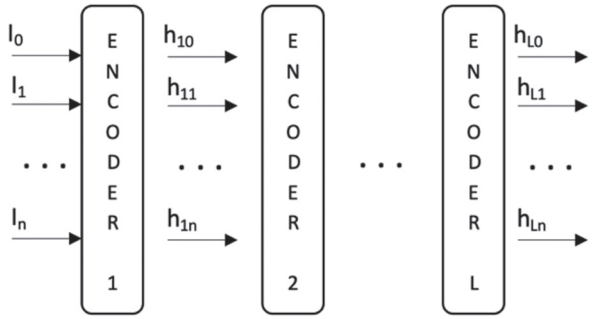


Fig. 5. Layers of the model

The first vector H of the final H representation is the basis for classification task. Two sub-processes form each encoder layer. As the first step, the inputs to the encoder are passed through a multi-head self-attention layer. This layer uses a series of matrix manipulations to extract language information from the data inputs. The definition of this process is multi-head self-attention (Fig. 6).

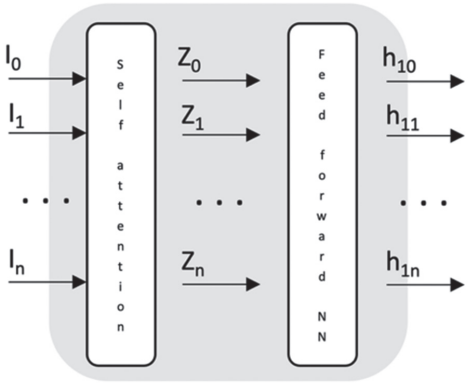


Fig. 6. The encoder dissection

The second sub-layer receives the outputs of the first layer after these outputs are residually connected and normalized. The second sub-layer retrieves the most valuable information from the attention layer. This data is residually connected and normalized again. After this, the outputs are sent onwards to the following encoder layer.

3. Theoretical propositions

This section is dedicated to proposition of a model for classifying the last phrase of three-turn conversations into any of pre-defined emotions. These emotions are Happy, Angry, Sad and Others. The context is defined by previous two sentences. The datasets are classified beforehand by the emotion they represent, since model training should be performed on high-quality data.

This task is solved by creation of neural network architecture. BERT is used as a state of art language representation model. The proposed model and BERT are compared in terms of their performance on the problem.

Table 1

Data sample

id	Turn 1	Turn 2	Turn 3	
112	It's cool meme	Ha-ha, yes	☺	Happy
154	I don't think it is a good idea	You better think twice	No, I won't!	Angry
186	In my hometown there are many chestnuts	Where is it?	In Kyiv	Others

The pre-processing is divided into two steps. The first one is data cleaning, which stands for defining and fixing data irregularities. The second one is tokenization, which stands for mathematical mapping of text data into a suitable input for classification models.

Data cleaning provides an ability to clarify meaning in the text and boost the coverage. Coverage is measured by pre-trained embeddings as the number of words for which pre-trained embeddings exist divided by the total amount of unique words in the problem. Since our dictionary is problem dependent and GloVe embedding matrix is generalizable, it is not certain that the embedding matrix has a vector representation of a given word. Thus, data cleaning is recurrently applied to boost coverage.

In our problem, data cleaning is used to transform abbreviations into their initial state and replace symbol emojis with Unicode emojis. This allows to improve sentence representation in terms of meaning.

After data cleaning, tokenization is applied. The words are vectorized and prepared for passing into classification models. The words are separated and mapped to unique numbers to decode them later. Keras, which is used to solve the problem, has this functionality built into the library.

As a result, tokenized texts are mapped with embedding matrices which represents the sequences in vector space. The vector represented sequences are passed to LSTM models. The input must all be of equal length to allow to train the model weights to recognize features at the same places in different sequences. To solve this, the vectors are zero padded to the same length, which is constrained to 100 words per sentence.

The Figure 7 illustrates the word length distribution is the training data.

Each sentence in a conversation is a matrix with elements of padded vectors representing the sentences. The corresponding labels are one-hot vectors. The index of the element 2 represents what emotion the training example is connected to. The indexes are as follows: 0 – Angry, 1 – Sad, 2 – Happy, 3 – Others. The dataset is ready for processing.

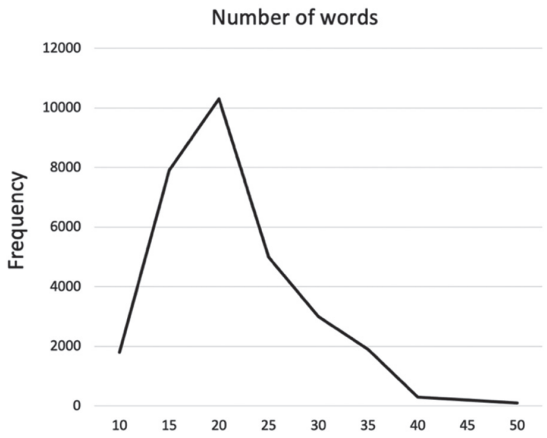


Fig. 7. Word lengths per sentence

1) Proposed model

The proposed model (Fig. 8) is built as follows. The standard LSTM outputs straight to a prediction layer and is filtered through a softmax function. The result is a probability vector. We use GloVe embeddings, trained on text corpus represented by Wikipedia data. The word embeddings of dimension 100 and LSTM layer of dimension 128 with built in dropout are used by the model.

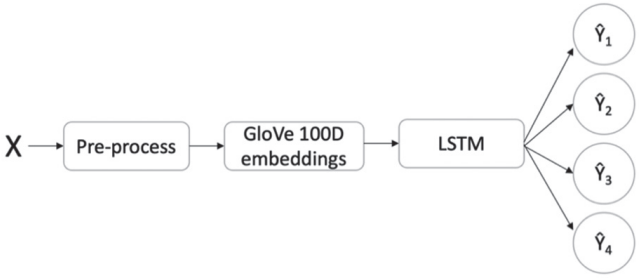


Fig. 8. The proposed model architecture

The model passes several improvements, which allow to define features with high impact on model performance. This impact is quantitively represented by F1 score. Such features are isolated and used to create simple networks.

The first improvement is to separate the phrases of the conversation and pass each one to LSTM nodes separately. Thus, the model is provided by a clearer vision of sentence differences.

The second improvement is to extract emojis and smileys from text for further separate processing. The emojis are represented with Emoji2Vec embeddings, which provides new language information to the model. Emoji2Vec is a 300-dimensional representation of Unicode smileys. Further concatenation of information contained in emojis with the data from the rest of the model is passed through a final dense layer. This allows the model to weight the relative importance of smileys in classifying the emotion. The rectified linear unit is the activation function in this dense layer.

BERT is used due to its applicability in wide range of problems. The model is hosted on TensorFlow hub. TensorFlow is a Google pre-trained machine learning repository.

It provides an ability to download an implementation which has to do with the following: the number of encoded layers, the number of heads in the multi-head attention, the number of dense layers in the feed forward network and the number of used training samples.

On the contrary to GloVe embeddings, BERT uses word-piece embeddings and follows different pre-processing and tokenizing which can be downloaded with the ready to use pre-trained model. This allows to not to perform pre-processing of the sentences but directly pass sentences to BERT.

4. Experimental results

The table 2 illustrates the comparative micro-averaged F1 results for the models on emotion classification. The performance of the models was considered by evaluation metric on Happy, Angry and Sad emotions. The model performance on emotion class Others was ignored in this evaluation.

The LSTM model achieved overall micro-averaged F1 as 0.616. This model required minimal data pre-processing and GloVe word embeddings. The word embeddings were trained on Wikipedia data. It can be seen in the table, that the model achieved the worst results on Happy conversations and the best results on Angry conversations. The shortcoming of the model is the inability to distinguish the Happy, Angry and Sad emotion classes from the emotion class Others.

Table 2

F_1 scores

Model	Emotion F_1			Micro F_1
	Happy	Sad	Happy	
LSTM	0.523	0.601	0.724	0.616
BERT	0.687	0.799	0.771	0.752
SS-LSTM	0.556	0.**818**	0.784	0.719
Proposed Model	**0.784**	0.767	**0.811**	0.787

BERT model achieved better results in macro F1 comparing to LSTM due to presence of extra context information. The Angry emotion class was predicted as 0.771 F1 points. The Sad emotion class was also predicted better in F1 points comparing to LSTM. The proposed model F1 scores are shown in the table. The model has shown the best results in Angry and Happy emotion class.

The confusion matrix of the proposed model is shown in table 3. The matrix illustrates the distribution between the emotion labels in the test dataset. The most confusion comes from distinguishing the Others emotion class from Happy, Angry and Sad emotion classes. Angry and Sad emotion classes are never evaluated by the model as Happy and vice versa. The confusion between predicting Angry and Sad labels is rarely present. On this basis, the

focus should be put on prediction improvement between Others emotion class and the other three emotion classes.

Table 3

Confusion Matrix

TRUE	Predicted			
	Others	Happy	Sad	Angry
Others	4103	101	80	94
Happy	63	215	4	1
Sad	40	0	212	8
Angry	46	0	6	334

The idea of whether the early stopping effectiveness on outfitting prevention is got by looking at the loss function over the training steps. The decrease of training loss is present over the training steps. The validation loss is decreasing by reaching the minimum after three epochs and rising afterwards (Fig. 9).

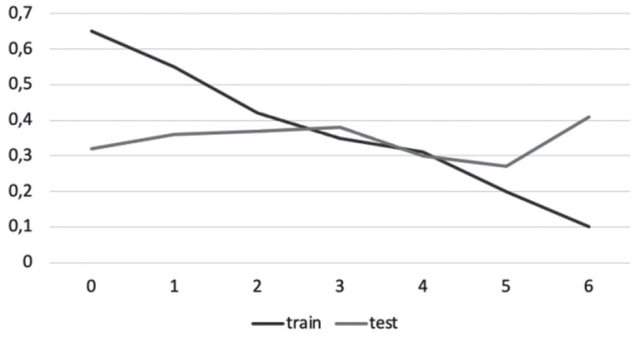


Fig. 9. Training and Validation Loss

The explanation behind this is the different distribution of emotions in the training dataset and validation dataset. This can be seen in table 4. The loss from the training set updates the gradients. The gradient updates change the validation loss in other ways. The amount of data points in the validation set is fewer comparing to the training set, which contributes to higher variance in the effects of updating the loss. On this basis, it is important to use early stopping after 3 epochs since the improvements in training loss are not translated to improvements in validation loss after this epoch.

Table 4

Emotion Class Distribution

	Others	Happy	Sad	Angry	Size
Train	.441	.98	.211	.194	29884
Validation	.793	.045	0.054	0.052	2467
Test	.882	0.039	0.061	0.044	5804

Conclusions

The problem of emotion classification is a complex task of language interpretation. The existing generic solutions for text data processing were considered in this work.

The evaluation of performance of the considered models was conducted. The proposed model shows better results comparing to generalized model with transfer learning.

There are many decisions applicable for solving the emotion classification problem. There is a variety of factors to consider when choosing the word embeddings and training methods to design the model architecture.

To date, machine learning models did not achieve the human performance in terms of language representation and emotion classification. The confusion is present in labeling complex sentence structures. Such nuances are explained by the language structure and its dynamic nature.

However, the area of capturing the language nuances is constantly evolving and new approaches are created every day.

References:

[1] *Alex Graves and Jürgen Schmidhuber*. Framewise phoneme classification with bidirec- tional lstm and other neural network architectures. Neural Networks, 18(5):602 – 610, 2005.

[2] *Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov.* Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15:1929–1958, 2014.

[3] *Y. Daniiel, K. Onyshchenko*. Implementation of Recursive Deep Learning Algorithms for Natural Language Processing. Information Systems and Technologies 2021, 2021.

[4] *I. Afanasieva, N. Golian, O. Hnatenko, Y. Daniiel, K. Onyshchenko.* Data exchange model in the internet of things concept. Telecommunications and Radio Engineering 78 (10), 2019.

[5] *Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa.* Natural language processing (almost) from scratch. CoRR, 2011.

[6] *Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov.* Bag of tricks for efficient text classification. CoRR, 2016.

[7] *Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita.* A survey of the usages of deep learning in natural language processing. CoRR, 2018.

[8] *James Allen*. Natural Language Understanding (2Nd Ed.). Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1995.

[9] *Umang Gupta, Ankush Chatterjee, Radhakrishnan Srikanth, and Puneet Agrawal.* A Sentiment-and-Semantics-Based Approach for Emotion Detection in Textual Conversa- tions. 2017.

[10] *Daniel Jurafsky and James H. Martin*. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000.

[11] *Christopher D.* Manning and Hinrich Schütze. Foundations of Statistical Natural Language Processing. MIT Press, Cambridge, MA, USA, 1999.

The article was delivered to editorial stuff on the 27.09.2021

УДК 519.7:007.52; 519.711.3 DOi 10.30837/ bi.2021.2(97).04

Д.М. Мешков[1], И.Д. Вечирская[2], Н.Е. Русакова[3]

[1]Студент, ХНУРЭ, Украина, dmytro.mieshkov@nure.ua

[2]Кандидат технических наук, доцент кафедры информатики,
Харьковский национальный университет радиоэлектроники, Украина,
iryna.vechirska@nure.ua, ORCID ID: 0000-0001-7964-2361

[3]Кандидат технических наук, доцент кафедры программной инженерии,
Харьковский национальный университет радиоэлектроники, Украина,
nataliia.rusakova@nure.ua, ORCID ID: 0000-0002-1242-6602

РЕШЕНИЕ ЗАДАЧИ ОПРЕДЕЛЕНИЯ ЗВАНИЯ СЛУЖАЩЕГО ВОЙСК УКРАИНЫ НА ОСНОВЕ АЛГЕБРЫ КОНЕЧНЫХ ПРЕДИКАТОВ

На сегодняшний день аппарат алгебры конечных предикатов развит настолько, что с его помощью можно описывать и моделировать всевозможные процессы человеческой интеллектуальной деятельности, начиная от описания решения простейших уравнений и заканчивая описанием языковых процессов. В данной статье рассматривается построение логической сети для решения задачи определения звания военнослужащего, которое зависит от ранга, типа звания, типа службы и других параметров. Для описания процесса распределения званий использован язык алгебры конечных предикатов. Используя метод формульной записи отношений построены парадигматические таблицы и описаны все звания военнослужащих, сформированы двудольные графы и построена соответствующая им логическая реляционная сеть. Каждую логическую сеть можно реализовать программно или преобразовать в электронную схему для автоматического решения того или иного класса задач, описанных моделью, для которой построена данная схема. Программная реализация доказала эффективность построенной сети.

ВОЕННОЕ ЗВАНИЕ, РАНГ, АЛГЕБРА КОНЕЧНЫХ ПРЕДИКАТОВ, ДВУДОЛЬНЫЙ ГРАФ, ЛОГИЧЕСКАЯ СЕТЬ

Мєшков Д.М., Вечирська І.Д., Русакова Н.Є. Вирішення задачі визначення звання військовослужбовця на основі алгебри кінцевих предикатів. На сьогодні апарат алгебри кінцевих предикатів розвинений настільки, що за його допомогою можна описувати та моделювати будь-які процеси людської інтелектуальної діяльності, починаючи від опису вирішення найпростіших рівнянь та закінчуючи описом мовних процесів. У статті розглядається побудова логічної мережі для вирішення задачі визначення звання військовослужбовця, яке залежить від рангу, типу звання, типу служби та інших параметрів. Для опису процесу розподілу звань використано мову алгебри кінцевих предикатів. Використовуючи метод формульного запису відносин побудовано парадигматичні таблиці та описано всі звання військовослужбовців, сформовані дводольні графи та побудовано відповідну їм логічну реляційну мережу. Кожну логічну мережу можна реалізувати програмно або перетворити на електронну схему для автоматичного вирішення того чи іншого класу завдань, описаних моделлю, для якої побудована дана схема. Програмна реалізація довела ефективність збудованої мережі.

ВІЙСЬКОВА ЗВАННЯ, РАНГ, АЛГЕБРА СКІНЧЕННИХ ПРЕДИКАТІВ, ДВОДОЛЬНИЙ ГРАФ, ЛОГІЧНА МЕРЕЖА

Meshkov D., Vechirska I., Rusakova N. Solution of the problem of determining the title of a service center of Ukraine on the basis of the algebra of finite predicates. Today, the apparatus of the algebra of finite predicates is so developed that it can be used to describe and simulate all kinds of processes of human intellectual activity, from describing the solution of the simplest equations to describing linguistic processes. This article discusses the construction of a logical network for solving the problem of determining the rank of a serviceman, which depends on the rank, type of rank, type of service and other parameters. The language of the algebra of finite predicates was used to describe the process of distributing ranks. Using the method of formulaic notation of relations, paradigmatic tables were built and all the ranks of military personnel were described, bipartite graphs were formed and a logical relational network corresponding to them was built. Each logical network can be implemented in software or converted into an electronic circuit for the automatic solution of one or another class of problems described by the model for which this circuit is built. The software implementation has proven the effectiveness of the constructed network.

MILITARY TITLE, RANK, ALGEBRA OF FINITE PREDICATES, TWO-SIDED GRAPH, LOGICAL NETWORK

Введение

В данной работе рассматривается подход к решению задачи определения звания служащего на основе алгебры конечных предикатов. Представить математическую модель определения звания служащего позволила методика построения логических сетей. Для этого необходимо было четко выделить основные узлы и их параметры, учитывая возможные связи как между узлами, так и между параметрами, провести анализ всех полученных отношений, определить предметные области каждой характеристики.

Задача – осуществить обработку символьных данных выделенных параметров на основе аппарата алгебры конечных предикатов [1, 2] с помощью

реляционной логической сети. Для разработки была выбрана область определения званий служащих в войсках Украины. Все значения предметных переменных были формализованы и представлены числовыми символами.

Рассматривались параметры особенностей условий службы человека для того, чтобы иметь представление его положении в служебной иерархии.

1. Математическое описание распределения званий в войсках Украины

1.1. Определение звания по типу и рангу

Перейдем к формальному описанию процесса распределения званий. С данной целью введем такие необходимые предметные переменные, как: x_1 – тип звания, x_2 – наименование звания, x_3 – ранг звания, x_1^1 – армейское, x_1^2 – корабельное, x_2^1 – солдат, x_2^2 – старший солдат, x_2^3 – младший сержант, x_2^4 – сержант, x_2^5 – старший сержант, x_2^6 – старшина, x_2^7 – прапорщик, x_2^8 – старший прапорщик, x_2^9 – младший лейтенант, x_2^{10} – лейтенант, x_2^{11} - старший лейтенант, x_2^{12} – капитан, x_2^{13} – майор, x_2^{14} – подполковник, x_2^{15} – полковник, x_2^{16} – генерал-майор, x_2^{17} – генерал-лейтенант, x_2^{18} – генерал-полковник, x_2^{19} – генерал армии Украины, x_2^{20} – матрос, x_2^{21} – старший матрос, x_2^{22} – мичман, x_2^{23} – старший мичман, x_2^{24} – капитан-лейтенант, x_2^{25} – контр-адмирал, x_2^{26} – вице-адмирал, x_2^{27} – адмирал. x_3^1 – первый, x_3^2 – второй, x_3^3 – третий, x_3^4 – четвертый. Представим формульное описание рангов военнослужащих разного звания на языке алгебры предикатов в виде парадигматической таблицы (табл. 1).

Таблица 1

Ранги и звания военнослужащих

Ранги	Звания	Армейские	Корабельные
Первый	Солдат	$x_1^1x_2^1x_3^1=q_1$	–
	Матрос	–	$x_1^2x_2^{20}x_3^1=q_{23}$
	Младший сержант	$x_1^1x_2^3x_3^1=q_3$	–
	Прапорщик	$x_1^1x_2^7x_3^1=q_7$	–
	Мичман	–	$x_1^2x_2^{22}x_3^1=q_{25}$
	Младший лейтенант	$x_1^1x_2^9x_3^1=q_9$	$x_1^2x_2^9x_3^1=q_{20}$
	Майор	$x_1^1x_2^{13}x_3^1=q_{13}$	–
	Генерал-майор	$x_1^1x_2^{16}x_3^1=q_{16}$	–
	Контр-адмирал	–	$x_1^2x_2^{25}x_3^1=q_{28}$
Второй	Старший солдат	$x_1^1x_2^2x_3^2=q_2$	–
	Старший матрос	–	$x_1^2x_2^{21}x_3^2=q_{24}$
	Сержант	$x_1^1x_2^4x_3^4=q_4$	–
	Старший прапорщик	$x_1^1x_2^8x_3^2=q_8$	–
	Старший мичман	–	$x_1^2x_2^{23}x_3^2=q_{26}$
	Старший сержант	$x_1^1x_2^5x_3^2=q_5$	–
	Лейтенант	$x_1^1x_2^{10}x_3^2=q_{10}$	$x_1^2x_2^{10}x_3^2=q_{21}$
	Подполковник	$x_1^1x_2^{14}x_3^2=q_{14}$	–
	Генерал-лейтенант	$x_1^1x_2^{17}x_3^2=q_{17}$	–
	Вице-адмирал	–	$x_1^2x_2^{26}x_3^2=q_{29}$
Третий	Старший лейтенант	$x_1^1x_2^{11}x_3^1=q_{11}$	$x_1^2x_2^{11}x_3^3=q_{22}$
	Полковник	$x_1^1x_2^{15}x_3^3=q_{15}$	–
	Генерал-полковник	$x_1^1x_2^{18}x_3^3=q_{18}$	–
	Адмирал	–	$x_1^2x_2^{27}x_3^3=q_{30}$
Четвертый	Старшина	$x_1^1x_2^6x_3^4=q_6$	–
	Капитан	$x_1^1x_2^{12}x_3^4=q_{12}$	–
	Капитан-лейтенант	–	$x_1^2x_2^{24}x_3^4=q_{27}$
	Генерал армии Украины	$x_1^1x_2^{19}x_3^4=q_{19}$	–

Далее введем переменную m – определение звания по типу и наименованию. Для определения m необходимо учесть признаки x_1, x_2 и x_3. Тогда предикат определения звания будет иметь вид:

$$m(x_1,x_2,x_3)=x_1^1x_2^1x_3^1\vee x_1^2x_2^{20}x_3^1\vee x_1^1x_2^3x_3^1\vee x_1^1x_2^7x_3^1\vee$$
$$x_1^2x_2^{22}x_3^1\vee x_1^1x_2^9x_3^1\vee x_1^1x_2^{13}x_3^1\vee x_1^1x_2^{16}x_3^1\vee x_1^2x_2^{25}x_3^1\vee$$
$$x_1^1x_2^2x_3^2\vee x_1^2x_2^{21}x_3^2\vee x_1^1x_2^4x_3^2\vee x_1^1x_2^8x_3^2\vee x_1^2x_2^{23}x_3^2\vee$$
$$x_1^1x_2^5x_3^2\vee x_1^1x_2^{10}x_3^2\vee x_1^1x_2^{14}x_3^2\vee x_1^1x_2^{17}x_3^2\vee x_1^1x_2^{26}x_3^2\vee$$
$$x_1^1x_2^{11}x_3^3\vee x_1^2x_2^{11}x_3^3\vee x_1^1x_2^{15}x_3^3\vee x_1^1x_2^{18}x_3^3\vee x_1^2x_2^{27}x_3^3\vee$$
$$x_1^1x_2^6x_3^4\vee x_1^1x_2^6x_3^4\vee x_1^1x_2^{12}x_3^4\vee x_1^2x_2^{24}x_3^4\vee x_1^1x_2^{19}x_3^4$$

1.2. Выполнение операции почленной дизъюнкции

$$x_1^1x_2^1x_3^1=q_1=m_1,\ x_1^1x_2^2x_3^2=q_2=m_2,\ x_1^1x_2^3x_3^1=q_3=m_3,$$
$$x_1^1x_2^4x_3^4=q_4=m_4,\ x_1^1x_2^5x_3^2=q_5=m_5,$$
$$x_1^1x_2^6x_3^4=q_6=m_6,\ x_1^1x_2^7x_3^1=q_7=m_7,$$
$$x_1^1x_2^8x_3^2=q_8=m_8,\ x_1^1x_2^9x_3^1\vee x_1^2x_2^9x_3^1=q_9\vee q_{20}=m_9,$$

$$x_1^1 x_2^{10} x_3^2 \vee x_1^2 x_2^{10} x_3^2 = q_{10} \vee q_{21} = m_{10},$$
$$x_1^1 x_2^{11} x_3^3 \vee x_1^2 x_2^{11} x_3^3 = q_{11} \vee q_{22} = m_{11},$$
$$x_1^1 x_2^{12} x_3^4 = q_{12} = m_{12},\ x_1^1 x_2^{13} x_3^1 = q_{13} = m_{13},$$
$$x_1^1 x_2^{14} x_3^2 = q_{14} = m_{14},\ x_1^1 x_2^{15} x_3^3 = q_{15} = m_{15},$$
$$x_1^1 x_2^{16} x_3^1 = q_{16} = m_{16},\ x_1^1 x_2^{17} x_3^2 = q_{17} = m_{17},$$
$$x_1^1 x_2^{18} x_3^3 = q_{18} = m_{18},\ x_1^1 x_2^{19} x_3^4 = q_{19} = m_{19},$$
$$x_1^2 x_2^{20} x_3^1 = q_{23} = m_{20},\ x_1^2 x_2^{21} x_3^2 = q_{24} = m_{21},$$
$$x_1^2 x_2^{22} x_3^1 = q_{25} = m_{22},\ x_1^2 x_2^{23} x_3^2 = q_{26} = m_{23},$$
$$x_1^2 x_2^{24} x_3^1 = q_{27} = m_{24},\ x_1^2 x_2^{25} x_3^1 = q_{28} = m_{25},$$
$$x_1^2 x_2^{26} x_3^2 = q_{29} = m_{26},\ x_1^2 x_2^{27} x_3^3 = q_{30} = m_{27}$$

Родственными называются такие выражения, которые после выполнение над ними операции почленной дизъюнкции приводят к равенству с левой частью в виде логического произведения, каждый сомножитель которого зависит только от одной предметной переменной. Мотивом, что побуждал выполнить операцию почленной дизъюнкции, есть стремление получить экономную систему воздействий определений.

1.3. Бинаризация предиката, связующего m с переменными

x_1, x_2 и x_3.

$$P_1(x_1, m) = x_1^1(m_1 \vee m_3 \vee m_4 \vee m_7 \vee m_8 \vee m_{10} \vee m_{12} \vee m_{13} \vee m_{15} \vee m_{17} \vee m_{18} \vee m_{20} \vee m_{21} \vee m_{22} \vee m_{24} \vee m_{25} \vee m_{27}) \vee x_1^2(m_2 \vee m_5 \vee m_9 \vee m_{11} \vee m_{14} \vee m_{19} \vee m_{23} \vee m_{26}) \vee (x_1^1 \vee x_1^2)(m_6 \vee m_{16} \vee m_{20})$$

$$P_2(x_2, m) = x_2^1 m_1 \vee x_2^2 m_2 \vee x_2^3 m_3 \vee x_2^4 m_4 \vee x_2^5 m_5 \vee x_2^6 m_6 \vee x_2^7 m_7 \vee x_2^8 m_8 \vee x_2^9 m_9 \vee x_2^{10} m_{10} \vee x_2^{11} m_{11} \vee x_2^{12} m_{12} \vee x_2^{13} m_{13} \vee x_2^{14} m_{14} \vee x_2^{15} m_{15} \vee x_2^{16} m_{16} \vee x_2^{17} m_{17} \vee x_2^{18} m_{18} \vee x_2^{19} m_{19} \vee x_2^{20} m_{20} \vee x_2^{21} m_{21} \vee x_2^{22} m_{22} \vee x_2^{23} m_{23} \vee x_2^{24} m_{24} \vee x_2^{25} m_{25} \vee x_2^{26} m_{26} \vee x_2^{27} m_{27}$$

$$P_3(x_3, m) = x_3^1(m_1 \vee m_3 \vee m_7 \vee m_9 \vee m_{13} \vee m_{16} \vee m_{20} \vee m_{22} \vee m_{25}) \vee x_3^2(m_2 \vee m_4 \vee m_5 \vee m_8 \vee m_{10} \vee m_{14} \vee m_{17} \vee m_{21} \vee m_{23} \vee m_{26}) \vee x_3^3(m_{11} \vee m_{15} \vee m_{18} \vee m_{27}) \vee x_3^4(m_6 \vee m_{12} \vee m_{19} \vee m_{24})$$

Ниже приведены двудольные графы для связи звания военнослужащего с его типом (рис. 1–3).

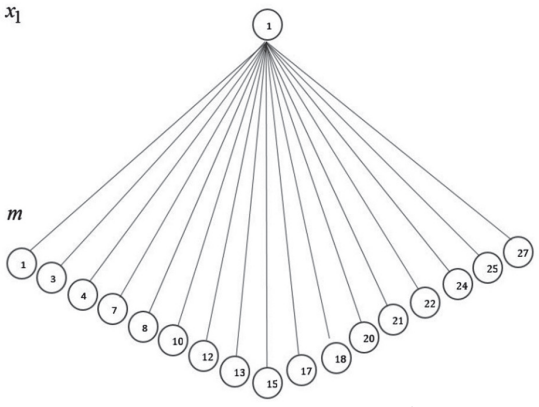


Рис. 1. Граф связи переменных x_1^1 и m

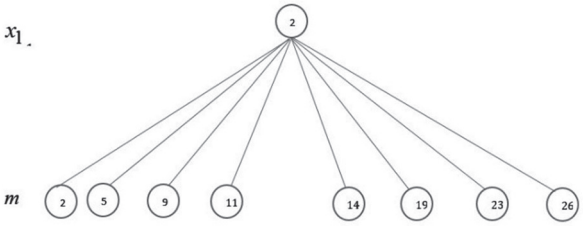


Рис. 2. Граф связи переменных x_1^2 и m

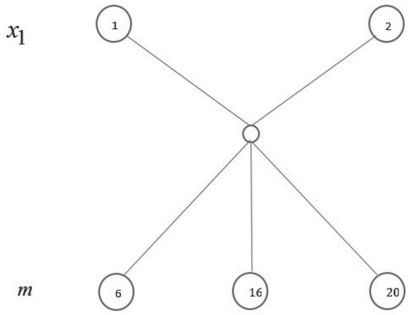


Рис. 3. Граф связи переменных $x_1^1 \vee x_1^2$ и m

На рис. 4–6 приведены двудольные графы, описывающие связь звания с рангом военнослужащего (первый, второй, третий, четвертый).

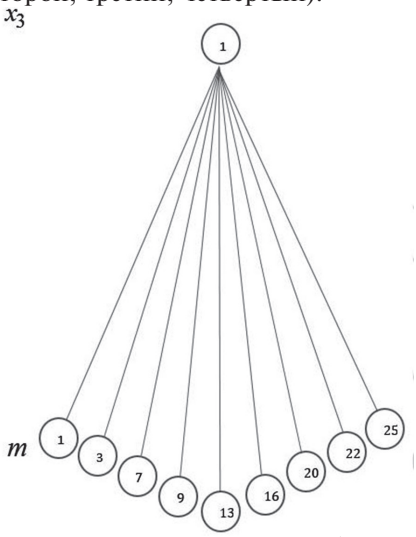


Рис. 4. Граф связи переменных x_3^1 и m

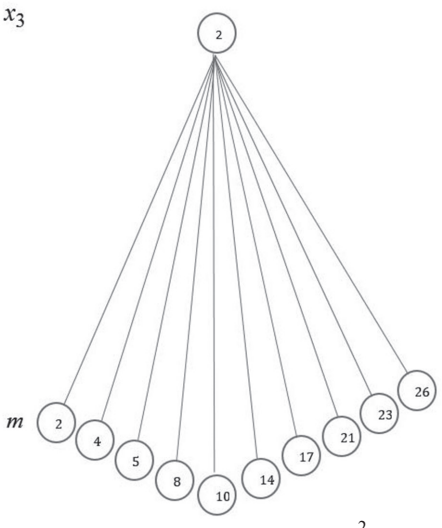


Рис. 5. Граф связи переменных x_3^2 и m

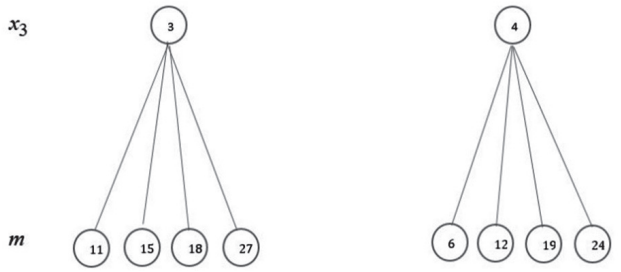


Рис. 6. Графы связи переменных x_3^3 и m, x_3^4 и m

2. Классификация по типу службы и состава.

Добавим следующий свойства в систему:

y_1 – тип службы, y_2 – тип состава; y_1^1 – срочная, y_1^2 – контрактная; y_2^1 – солдатский состав, y_2^2 – сержантский состав, y_2^3 – состав прапорщиков, y_2^4 – младший офицерский состав, y_2^5 – старший офицерский состав, y_2^6 – высший офицерский состав.

Тогда предикат определения звания по типу службы и состава будет выглядеть так:

$$Y = y_1^1 y_2^1 \vee y_1^1 y_2^2 \vee y_1^1 y_2^3 \vee y_1^2 y_2^1 \vee y_1^2 y_2^2 \vee$$
$$y_1^2 y_2^3 \vee y_1^2 y_2^4 \vee y_1^2 y_2^5 \vee y_1^2 y_2^6$$

Парадигматическая таблица определения звания представлена в табл. 2.

Таблица 2

Определение звания по типу службы военнослужащего

	Срочная	Контрактная
Солдатский состав	$y_1^1 y_2^1 = k_1$	$y_1^2 y_2^1 = k_4$
Сержантский состав	$y_1^1 y_2^2 = k_2$	$y_1^2 y_2^2 = k_5$
Состав прапорщиков	$y_1^1 y_2^3 = k_3$	$y_1^2 y_2^3 = k_6$
Младший офицерский состав	–	$y_1^2 y_2^4 = k_7$
Старший офицерский состав	–	$y_1^2 y_2^5 = k_8$
Высший офицерский состав	–	$y_1^2 y_2^6 = k_9$

Введем замену n – определение звания по типу службы и типу состава:

$$n(y_1, y_2) = y_1^1 y_2^1 \vee y_1^1 y_2^2 \vee y_1^1 y_2^3 \vee y_1^2 y_2^1 \vee y_1^2 y_2^2 \vee$$
$$y_1^2 y_2^3 \vee y_1^2 y_2^4 \vee y_1^2 y_2^5 \vee y_1^2 y_2^6$$

Выполним операцию почленной дизъюнкции и сформируем соответсвенные зависимости:

$$y_1^1 y_2^1 \vee y_1^2 y_2^1 = k_1 \vee k_4 = n_1,\ y_1^1 y_2^2 \vee y_1^2 y_2^2 = k_2 \vee k_5 = n_2,$$
$$y_1^1 y_2^3 \vee y_1^2 y_2^3 = k_3 \vee k_6 = n_3,\ y_1^2 y_2^4 = k_7 = n_4,$$
$$y_1^2 y_2^5 = k_8 = n_5,\ y_1^2 y_2^6 = k_9 = n_6$$

Проведем бинаризацию:

$$P_4(y_1, n) = (y_1^1 \vee y_1^2)(n_1 \vee n_2 \vee n_3) \vee y_1^2 (n_4 \vee n_5 \vee n_6)$$
$$P_5(y_2, n) = y_2^1 n_1 \vee y_2^2 n_2 \vee y_2^3 n_3 \vee y_2^4 n_4 \vee y_2^6 n_5 \vee y_2^6 n_6$$

Двудольные графы, соответствующие описанным связям представлены на рис. 7–9.

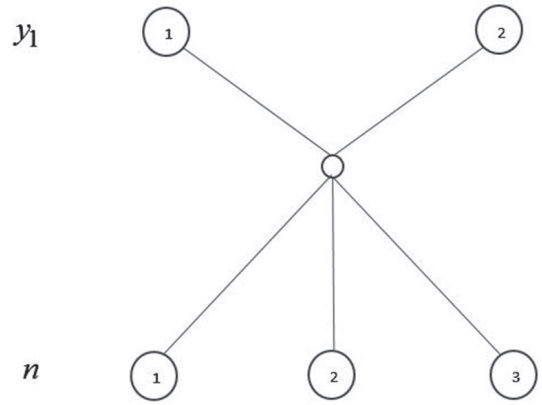


Рис. 7. Граф связи переменных $y_1^1 \vee y_1^2$ и n

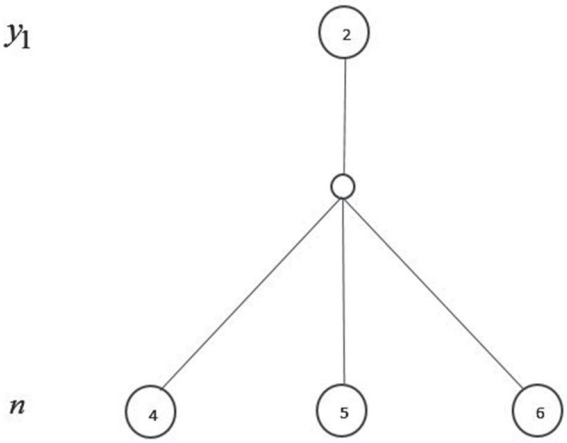


Рис. 8. Граф связи переменных y_1^2 и n

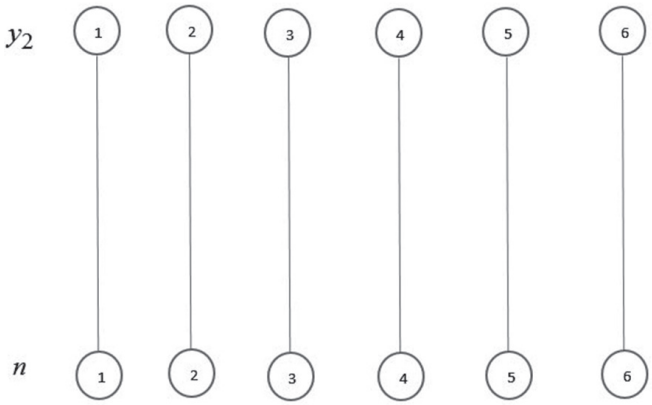


Рис. 9. Граф связи переменных y_2 и n

3. Математическая модель определения звания по типу звания, рангу и типу состава.

Для описания зависимостей между званием, его типом (армейское, корабельное), рангом и типом состава (солдат, сержант, прапорщик и т.д.) была составлена парадигматическая таблица (табл. 3), в которой формулами алгебры предикатов описаны эти связи [4].

Таблица 3

Парадигматическая таблица связи звания с его типом и рангом

	Звания	**Тип состава**					
		Солдат	**Сержант**	**Прапорщик**	**Младший офицерский состав**	**Старший офицерский состав**	**Высший офицерский состав**
Армейские	Солдат	$x_1^1 x_2^1 y_2^1$	–	–	–	–	–
	Старший солдат	$x_1^1 x_2^2 y_2^1$	–	–	–	–	–
	Младший сержант	–	$x_1^1 x_2^3 y_2^2$	–	–	–	–
	Сержант	–	$x_1^1 x_2^4 y_2^2$	–	–	–	–
	Старший сержант	–	$x_1^1 x_2^5 y_2^2$	–	–	–	–
	Главный сержант	–	$x_1^1 x_2^6 y_2^3$	–	–	–	–
	Прапорщик	–	–	$x_1^1 x_2^7 y_2^3$		–	–
	Старший прапорщик	–	–	$x_1^1 x_2^8 y_2^3$	–	–	–
	Младший лейтенант	–	–	–	$x_1^1 x_2^9 y_2^4$	–	–
	Лейтенант	–	–	–	$x_1^1 x_2^{10} y_2^4$		–
	Старший лейтенант	–	–	–	$x_1^1 x_2^{11} y_2^4$	–	–
	Капитан	–	–	–	$x_1^1 x_2^{12} y_2^4$	–	–
	Майор	–	–	–	–	$x_1^1 x_2^{13} y_2^5$	–
	Подполковник	–	–	––	–	$x_1^1 x_2^{14} y_2^5$	–
	Полковник	–	–	–	–	$x_1^1 x_2^{15} y_2^5$	–
	Генерал-майор	–	–	–	–	–	$x_1^1 x_2^{16} y_2^6$
	Генерал-лейтенант	–	–	–	–	–	$x_1^1 x_2^{17} y_2^6$
	Генерал-полковник	–	–	–	–	–	$x_1^1 x_2^{18} y_2^6$
	Генерал Армии Украины	–	–	–	–	––	$x_1^1 x_2^{19} y_2^6$
Корабельные	Матрос	$x_1^2 x_2^{20} y_2^1$	–	–	–	–	–
	Старший матрос	$x_1^1 x_2^{21} y_2^1$	–	–	–	–	–
	Старшина	–	$x_1^2 x_2^{22} y_2^2$	–	–	–	–
	Мичман	–	–	$x_1^2 x_2^{23} y_2^3$	–	–	–
	Старший мичман	–	–	$x_1^2 x_2^{24} y_2^3$	–	–	–
	Младший лейтенант	–	–	–	$x_1^2 x_2^9 y_2^4$	–	––
	Лейтенант	–	–	–	$x_1^2 x_2^{10} y_2^4$	–	–
	Старший лейтенант	–	–	–	$x_1^2 x_2^{11} y_2^4$	–	–
	Капитан	–	–	–	$x_1^2 x_2^{12} y_2^4$	$x_1^2 x_2^{12} y_2^5$	–
	Контр-адмирал	–	–	–	–	–	$x_1^2 x_2^{25} y_2^6$
	Вице-адмирал	–	–	–	–	–	$x_1^2 x_2^{26} y_2^6$
	Адмирал	–	–	–	–	–	$x_1^2 x_2^{27} y_2^6$

Производим бинаризацию функции-номера ячейки таблицы.

$$P_6(x_1,x_2,y_2)=x_1^1y_2^1(x_2^1\vee x_2^2)\vee x_1^1y_2^2(x_2^3\vee x_2^4\vee x_2^5\vee x_2^6)\vee$$
$$x_1^1y_2^3(x_2^7\vee x_2^8)\vee(x_1^1\vee x_1^2)\vee y_2^4(x_2^9\vee x_2^{10}\vee x_2^{11}\vee x_2^{12})\vee$$
$$x_1^1y_2^5(x_2^{13}\vee x_2^{14}\vee x_2^{15})\vee x_1^1y_2^6(x_2^{16}\vee x_2^{17}\vee x_2^{18}\vee x_2^{19})\vee$$
$$x_1^2y_2^1(x_2^{20}\vee x_2^{21})\vee x_1^2y_2^2x_2^{22}\vee x_1^2y_2^3(x_2^{23}\vee x_2^{24})\vee$$
$$x_1^2y_2^5x_2^{12}\vee x_1^2y_2^6(x_2^{25}\vee x_2^{26}\vee x_2^{27})$$

Ниже приведена логическая сеть для определения звания служащего войск Украины (рис. 10).

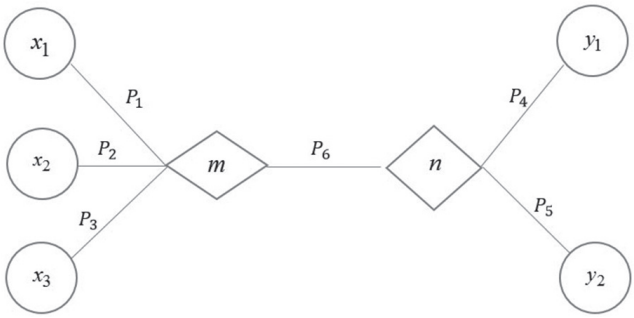


Рис. 10. Логическая сеть

Данная логическая сеть позволяет, принимая как входные данные тип звания, тип состава и ранг звания как необходимые, а также тип службы как дополнительные данные, находить наименование звания служащего.

4. Программная реализация

Была разработана программная реализация логической сети, с помощью которой выбирая из предложенных вариантов входных данных можно узнать наименование звания по вводимым атрибутам (рис. 11–14).

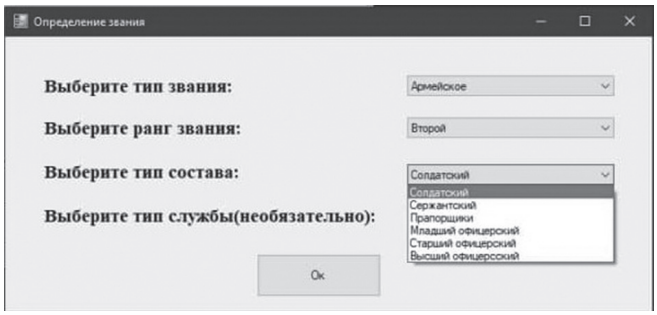


Рис. 11. Вид главной формы программы

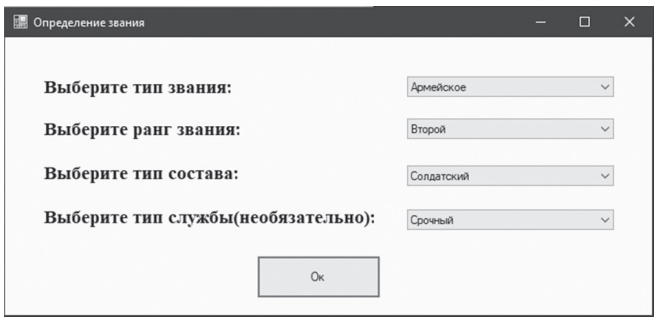


Рис. 12. Основной функционал

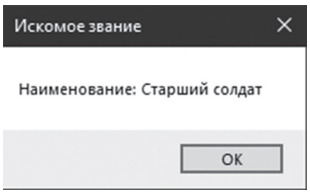


Рис. 13. Результат работы программы

Программа является защищенной от некорректного ввода: пользователь не вводит данные самостоятельно, а лишь выбирает из предложенных. При выборе несовместимых параметров (к примеру, солдатский состав и четвертый ранг) программа сообщит, что искомого звания (с такими параметрами) не существует.

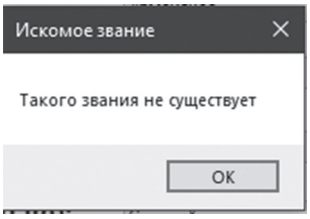


Рис. 14. Информация о неверном вводе данных

Выводы

В результате анализа закона Украины «О всеобщей воинской обязанности и воинской службе» и математического моделирования с применением аппарата алгебры конечных предикатов в статье были математически описаны все звания военнослужащих Украины, которые зависят от ранга, типа состава, типа службы. Построена логическая сеть для описания задачи определения звания военнослужащего. Каждой модели соответствует своя логическая сеть, которая состоит из полюсов и ветвей. Каждому из полюсов сети ставится в соответствие свой атрибут. В любой момент времени полюс несет знание о значении своего атрибута. Указав состояние каждого из полюсов, получаем состояние сети в данный момент времени. Разработанная сеть была реализована в виде программного приложения и показала свою результативность на различной выборке данных.

Список литературы:

[1] M.F. Bondarenko, N.P. Kruglikova, I.O. Leshchynska, N.E. Rusakova, Yu.P. Shabanov-Kushnarenko, About algebra of predicates // Bionics of Intelligence: Sci. Mag. – 2010. – № 3 (74). – P. 3–7.

[2] I.D. Vechirska, I.E. Goncharov, T.M. Khamitov, Bulding of logic network for the diagnosis and management of emergencies // Bionics of Intelligence: Sci. Mag. – 2015. – № 2 (85). – P.41–51.

[3] Bondarenko M.F., Shabanov-Kushnarenko Yu.P. Mozgopodobnyie strukturyi: spravochnoe posobie. Tom pervyiy. [Tekst] – K.: Naukova dumka, 2011. – 460 s.

[4] Zakon Ukrainyi ot 25 marta 1992 g. # 2232-XII «O vseobschey voinskoy obyazannosti i voennoy sluzhbe»

Поступила в редколлегию 12.10.2021

УДК 004.8 DOi 10.30837/ bi.2021.2(97).05

В.М. Горбенко[1], К.Г. Онищенко[2], І.В.Афанасьєва[3], Р.В. Каменєв[4]

[1]Студент, кафедра програмної інженерії,
Харківський національний університет радіоелектроніки,
vladyslav.horbenko@nure.ua, ORCID iD: 0000-0002-4988-3268

[2]Асистент кафедри Програмної інженерії, Харківський національний університет радіоелектроніки,
kostiantyn.onyshchenko@nure.ua, ORCID iD: 0000-0002-7746-4570

[3]Доцент кафедри програмної інженерії, кандидат технічних наук,
Харківський національний університет радіоелектроніки,
iryna.afanasieva@nure.ua, ORCID iD: 0000-0003-4061-0332

[1]Студент, кафедра програмної інженерії, Харківський національний університет радіоелектроніки,
roman.kameniev@nure.ua, ORCID iD: 0000-0002-8263-8862

АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ТА МОДЕЛЕЙ ГЛИБИННОГО НАВЧАННЯ В ЗАДАЧАХ ОБРОБКИ ПРИРОДНОЇ МОВИ

Розглянуто підходи та методи глибинного навчання природної мови. Глибинні нейронні мережі створюються як мережі прямого поширення, але дослідження дуже успішно застосували рекурентні нейронні мережі до таких задач, як моделювання мов. Згорткові глибинні нейронні мережі застосовуються в комп'ютерному зорі та природній мові. Згорткові глибинні нейронні мережі також було застосовано до акустичного моделювання для автоматичного розпізнавання мовлення. Гнучка нейронна мережа, дані функції використовуються для описання прямокутних об'єктів у вигляді двійкової маски в природній мові, а також для обрахування масштабної регресії для покращення точності визначеного положення. Багатозмінна логістична функція, що використовуються для описання прямокутних об'єктів у вигляді двійкової маски, а також для обрахування масштабної регресії для покращення точності визначеного положення. Глибинна мережа переконань, що є одним із типів глибинних нейронних мереж, що складений з декількох шарів прихованих змінних, що мають з'єднання, але не між вузлами всередині кожного шару. Нейронні мережі зберігання та вибірки великої пам'яті, її особливістю є те що вона є швидкою нейронною мережею з багатьма шарами які можуть використовуватись одночасно. Глибинне машинне навчання. Складені автокодувальники, ціль авдокодувальника, що продиктована поняттям «доброго представлення». Розглянуто існуючі моделі та методи для вирішення поставлених задач. Було проаналізовано різні методи та проведено порівняння, позитивних та негативних факторів вибору того чи іншого методу. В кінці проведені висновки, в яких було підсумковано як позитивні так і негативні сторони оглянутих методів в розрізі застосовування в природній мові.

ГЛИБИННЕ НАВЧАННЯ, ПРИРОДНОЇ МОВИ, ІНСТРУМЕНТИ МАШИННОГО НАВЧАННЯ, МАШИННЕ НАВЧАННЯ, НАВЧАННЯ З УЧИТЕЛЕМ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ

Горбенко В.Н., Онищенко К.Г., Афанасьева И.В., Каменев Р.В. Анализ существующих методов и моделей глубинного обучения в задачах обработки природного языка. Рассмотрены подходы и методы глубинного обучения естественному языку. Глубинные нейронные сети создаются как сети прямого распространения, но исследования очень успешно применили рекуррентные нейронные сети к таким задачам, как моделирование языков. Сверточные глубинные нейронные сети используются в компьютерном зрении и естественной речи. Сверточные глубинные нейронные сети также были применены к акустическому моделированию для автоматического распознавания речи. Гибкая нейронная сеть, данные функции используются для описания прямоугольных объектов в виде двоичной маски в естественной речи, а также для вычисления масштабной регрессии для улучшения точности определенного положения. Многосменная логистическая функция, используемая для описания прямоугольных объектов в виде двоичной маски, а также для вычисления масштабной регрессии для улучшения точности определенного положения. Глубинная сеть убеждений является одним из типов глубинных нейронных сетей, составленным из нескольких слоев скрытых переменных, имеющих соединение, но не между узлами внутри каждого слоя. Нейронные сети хранения и выборки большой памяти, ее особенностью является то, что она является быстрой нейронной сетью со многими слоями, которые могут использоваться одновременно. Глубинное машинное обучение. Составлены автокодировщики, цель авдокодировщика, продиктованная понятием «хорошего представления». Рассмотрены существующие модели и методы решения поставленных задач. Были проанализированы различные методы и проведены сравнения, положительных и негативных факторов выбора того или иного метода. В конце проведены выводы, в которых были итогированы как положительные так и отрицательные стороны осмотренных методов в разрезе применения в естественном языке.

ГЛУБИННОЕ ОБУЧЕНИЕ, ЕСТЕСТВЕННЫЙ ЯЗЫК, ИНСТРУМЕНТЫ МАШИННОГО ОБУЧЕНИЯ, МАШИННОЕ ОБУЧЕНИЕ, ОБУЧЕНИЕ С УЧИТЕЛЕМ, СВЕРТОЧНЫЕ НЕЙРОННЫЕ СЕТИ

Vladyslav Horbenko, Kostiantyn Onyshchenko, Iryna Afanasieva, Roman Kameniev. Analysis of existing methods and models of in-depth learning in the problems of natural language processing. Approaches and methods of deep learning of natural language are considered. Deep neural networks are being created as direct propagation networks, but research

has very successfully applied recurrent neural networks to tasks such as language modeling. Convolutional deep neural networks are used in computer vision and natural language. Convolutional deep neural networks have also been applied to acoustic modeling for automatic speech recognition. Flexible neural network, these functions are used to describe rectangular objects in the form of a binary mask in natural language, as well as to calculate scale regression to improve the accuracy of a particular position. Multivariate logistic function used to describe rectangular objects in the form of a binary mask, as well as to calculate scale regression to improve the accuracy of a particular position. A deep network of beliefs, which is a type of deep neural network made up of several layers of hidden variables that have connections but not between nodes within each layer. Neural networks store and sample large memory, its feature is that it is a fast neural network with many layers that can be used simultaneously. Deep machine learning. Compound car encoders, the purpose of the car encoder, which is dictated by the concept of «good performance». The existing models and methods for solving the tasks are considered. Different methods were analyzed and a comparison of positive and negative factors of choosing a method was made. In the end, the conclusions were summarized, which summarized both the positive and negative aspects of the reviewed methods in terms of application in natural language.

IN-DEPTH LEARNING, NATURAL LANGUAGE, MACHINE LEARNING TOOLS, MACHINE LEARNING, TEACHER LEARNING, CONVOLVED NEURAL NETWORKS.

Вступ

В наш час, коли інформаційний простір не обмежується територіально і для її отримання не потрібно багато часу, ми маємо велику кількість не перевіреної інформаціїї, що може містити пагубний контент на осіб, які психологічно слабкі до негативу, або особи, що знайомляться інформацією, є неповнолітніми. Тому потрібні інструменти які можуть опрацьовувати велику кількість текстової інформаціїї та знаходити в ньому харасмент однієї особи до іншої, домагання та расизм.

Обробка природної мови (NLP – Natural Language Processing) – це підрозділ інформаційних технологій, штучного інтелекту та лінгвістики, метою якого є вивчення проблем комп'ютерного аналізу та синтезу природної мови. Повне розуміння та відтворення сенсу мови – надзвичайно складне завдання, оскільки людська мова має цілий ряд особливостей.

Людська мова – це складний механізм, набутий в процесі еволюції, що використовує звук або писемність для передачі інформації від однієї людини до іншої або до групи людей. Мова це складна система слів, що у купі створює речення, а ті в свою чергу надають послідовну інформацію, яка сприймається опонентом і він може отримати інформацію, будь-якого характеру, це можуть бути знання, попередження, оповідь, будь-що.

Обробка природної мови (NLP – Natural Language Processing) – це підрозділ інформаційних технологій, штучного інтелекту та лінгвістики, метою якого є вивчення проблем комп'ютерного аналізу та синтезу природної мови. Повне розуміння та відтворення сенсу мови – надзвичайно складне завдання, оскільки людська мова має цілий ряд особливостей.

Технології машинного навчання все глибше потрапляють у наш життєвий побут і можуть використовуватись у таких продуктах як камери або смартфони. Машинне навчання використовується, щоб виділити об'єкти в зображенні, перетворити мовлення в текст або надати релевантні результати пошуку. Все частіше для вирішення данної задачі використовують таку галузь машинного навчання, як глибинне навчання. Воно стало на зміну звичайному машинному навчанню, що потребувало створення складних систем, які б перетворювали сирі дані у представлення, що може бути оброблене системою. Отже, глибинне навчання — це набір методів, що використовує багатошарові штучні нейронні мережі, щоб забезпечити найсучаснішу точність виконання завдань.

1. Постановка задачі

Необхідно розглянути методи глибинного навчання та їх використання в обробці природної мови, огляд позитивних та негативних сторін у порівнянні одного методу з іншим.

Різні архітектури глибинного навчання, такі як глибинні нейронні мережі, згорткові глибинні нейронні мережі, глибинні мережі переконань та рекурентні нейронні мережі застосовувалися в таких областях, як комп'ютерне бачення, автоматичне розпізнавання мовлення, обробка природної мови, розпізнавання звуків та біоінформатика, де вони, як було показано, представляють передові результати в різноманітних задачах.

В кінці зробимо висновки, в яких задачах вибрати той чи інший підхід, що він може нам запропонувати, та в яких ситуаціях може вирішити ту чи іншу задачу.

2. Огляд існуючих моделей та методів для вирішення поставленої задачі

Є дуже багато варіантів архітектури глибинного навчання. В основному, вони відгалужені від батьківських архітектур. Порівняння ефективності різної архітектури не завжди є можливим, адже не всі з них оцінювались на однакових вхідних наборах даних. Глибинне навчання на даний час, дуже швидко розвивається і нових варіантів або алгоритмів з'являються дуже швидко.

Глибинна нейронна мережа — це штучна нейронна мережа з декількома прихованими шарами вузлів між вхідним та вихідним шарами. Подібно до пласких штучних нейронних мереж, глибинні нейронні мережі можуть моделювати складні нелінійні відношення [6].

Архітектури глибинних нейронних мереж, наприклад, для виявлення об'єктів та граматичного аналізу,

породжують композиційні моделі, де об'єкт виражається як шарувата композиція примітивів зображення. Додаткові шари дозволяють композиції включати ознаки з нижчих шарів, забезпечуючи потенціал для моделювання складних даних меншою кількістю вузлів, ніж настільки ж ефективна пласка мережа.

Глибинні нейронні мережі створюються як мережі прямого поширення, але дослідження дуже успішно застосували рекурентні нейронні мережі, до таких задач, як моделювання мов. Згорткові глибинні нейронні мережі застосовуються в комп'ютерному зорі та природній мові. Згорткові глибинні нейронні мережі також було застосовано до акустичного моделювання для автоматичного розпізнавання мовлення.

Гнучка нейронна мережа, тренується за допомогою одного з стандартних алгоритму зворотного поширення, або іншими словами метод навчання багатошарового перцептрону. Основи безперервного зворотного поширення були створені в розрізі теорії керування Генрі Келлі та Артуром Брайсоном в 1960-1961 роках, принципи були застосовані в динамічному програмуванні. В 1962 році Стюарт Дрейфус розробив простіше рішення на основі ланцюгового правила, це було подано як багатоетапний метод оптимізації динамічної системи. Знаходження вагових коефіцієнтів відбувається стохастичним найшвидшим спуском, ітеративний метод оптимізації градієнтного спуску за допомогою стохастичного наближення, використовується таке рівняння [1].

$$w_{ij}(t+1) = w_{ij}(t) + \eta \frac{\partial C}{\partial w_{ij}},$$

де h – темп навчання, C – функція витрат. Вибір функції витрат буде залежить від того який тип навчання було використано: спонтанне – при вирішенні яких випробовувана система спонтанно навчається виконувати поставлене завдання, без втручання з боку експериментатора, з підкріплення, виконують програмні агенти в певному середовищі задля максимізації деякого уявлення про сукупну винагороду, кероване, в ході якого випробувана система примусово навчається за допомогою наявної множини прикладів «стимул-реакція» з метою визначення «реакції» для «стимулів», які не належать до наявної множини прикладів, та функції активації – залежність вихідного сигналу штучного нейрона від вхідного.

Багатозмінна логістична функція, функція розраховується за допомогою рівняння,

$$p_j = \frac{\exp(x_j)}{\sum_k \exp(x_k)},$$

де p_j – є ймовірністю класу, а x_j та x_k – сумарний вхід до вузлів j та k на одному й тому ж рівні.

Перехресна ентропія розраховується за формулою

$$C = -\sum_j d_j \log(p_j).$$

В даній формулі d_j – це цільові ймовірності для вихідних вузлів j, p_j – вихідна ймовірність для j після активаційної функції.

Дані функції використовуються для описання прямокутних об'єктів у вигляді двійкової маски в природній мові, а також для обрахування масштабної регресії для покращення точності визначеного положення.

Як із глибинним нейронним навчанням так із штучним нейронним навчанням, в процесі роботи може виникати багато проблем, якщо їх тренувати наївно. Основними проблемами є перенавчання та тривалість обчислення.

Глибинна мережа переконань – є одним із типів глибинних нейронних мереж, що складений з декількох шарів прихованих змінних, що мають з'єднання, але не між вузлами всередині кожного шару.

В ході тренування на наборі прикладів спонтанним чином, даний метод може навчитися краще відбудовувати свої входи. Кожний шар являє собою детектор ознак, набір методик, що дозволяє системі автоматично виявляти представлення, необхідні для виявлення ознак, на входах. Глибинна мережа переконань розглядається як суміш простих та спонтанних мереж, де попередній прихований шар слугує явним шаром наступному. Це веде до швидкої пошарової процедури спонтанного тренування, в якій порівняльна розбіжність застосовується до кожної підмережі по черзі, починаючи з найнижчої пари шарів.

Глибинна мережа переконань працює наступним чином:

1) натренувати обмежену машину Больцмана (рис. 1) на X, щоби отримати матрицю її вагових коефіцієнтів, W. Використовувати її як матрицю вагових коефіцієнтів між двома нижніми шарами мережі;

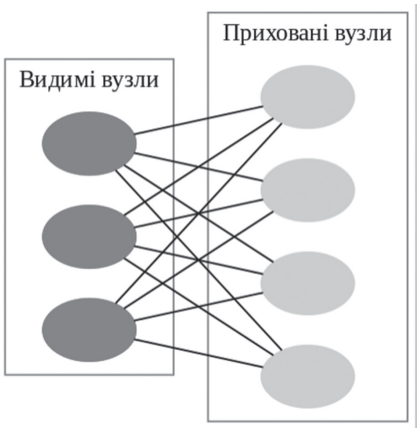


Рис. 1. Обмежена машина Больцмана

2) перетворити X за допомогою цієї ОМБ для отримання нових даних X', або шляхом вибірки, або обчисленням середньої активації прихованих вузлів;

3) повторювати цю процедуру з X ← X' для наступної пари шарів, поки не буде досягнуто двох верхніх шарів мережі;

4) здійснити тонке налаштування всіх параметрів цієї глибинної архітектури по відношенню до того,

що виконує роль логарифмічної правдоподібності ГМП, або по відношенню до критерію керованого тренування (після додавання додаткових механізмів навчання для перетворення навченого представлення на керовані передбачення, наприклад, лінійного класифікатора).

Як тільки обмежену машину Больцмана було натренеровано, на неї накладається інша, беручи свої входи із завершального вже натренованого рівня. Значенням нового вхідного видимого шару встановлюється тренувальний вектор, а значення вузлів уже натренованих шарів встановлюються із застосуванням поточних вагових коефіцієнтів та зсувів. Потім нова обмежена машинна Больцмана тренується за наведеною вище процедурою. Весь цей процес повторюється до досягнення бажаного критерію зупинки.

Хоча наближення порівняльної розбіжності до максимальної правдоподібності і є дуже грубим, було показано, що порівняльна розбіжність не слідує градієнтові будь-якої функції, було емпірично показано, що вона є ефективною в тренуванні глибинних архітектур.

Одним із нових досягнень у глибинному навчанні є застосування згорткових глибинних мереж переконань, що мають структуру, дуже схожу до згорткових нейронних мереж, і тренуються схоже до глибинних мереж переконань. Тому, вони використовують двовимірну структуру зображень, так само як це роблять згорткові глибинні мережі переконань, і використовують попереднє тренування, як глибинні мережі переконань. Загальна структура, що може застосовуватися в багатьох задачах обробки зображень та сигналів. за останній час, було багато результатів на стандартних наборах зображень, таких як CIFAR, було отримано із застосуванням згорткові глибинні мережі переконань.

Нейронні мережі зберігання та вибірки великої пам'яті – є одним із видів глибинного навчання, її особливістю є те що вона є швидкою нейронною мережею з багатьма шарами які можуть використовуватись одночасно. Фільтри можуть бути нелінійними, логічними та не стаціонарними і мають неперервне навчання.

Данна нейронна мережа слугує динамічною нейронною мережею в просторово-часовій області визначення та в обох. Дана швидкість забезпечується геббовими (це теорія в нейронауці, яка пропонує пояснення пристосування нейронів мозку під час процесу навчання, описуючи основний механізм синаптичної пластичності, в якому підвищення синаптичної дієвості виникає в результаті повторюваного й постійного стимулювання пресинаптичною клітиною постсинаптичної) ваговими коефіцієнтами з'єднань, що об'єднує між собою не схожі по своїй суті фільтрів, функції попередньої обробки, в її багатьох шарах, для динамічного ранжування ваги різних шарів та функцій по відношенню до заданої задачі глибинного навчання. Вона повністю прозорою, це досягається завдяки своїм ваговим коефіцієнтам з'єднань.

Метод нейронних мереж зберігання та вибірки великої пам'яті, застосовували в медичних та фінансових прогнозах та адаптивному фільтруванні зашумленого мовлення в невідомому шумі, розпізнаванні нерухомих зображень, безпеці програмного забезпечення, адаптивному керуванні нелінійними системами та ін. У порівняльному дослідженні з розпізнавання символів, дана методика, мала значно вищу швидкість обрахування і меншу похибку, в порівнянні із згортковою нейронною мережею на основі фільтрування функціями ReLU та максимальною підвибіркою.

Застосунки на базі даного методу показують приховані аспекти даних, що є прихованими від мереж поверхневого навчання, та навіть людської інтуїції, що покладається на очі та слух.

Метод нейронних мереж зберігання та вибірки великої пам'яті, був запропонований в 1996 році і вдосконалено Даніелем Ґраупе та Губертом Кордилевським.

Глибинне машинне навчання – це один із видів двійкового парного марковського випадкового поля з шарами латентних випадкових змінних [6]. Що є мережею симетрично спарованих випадкових двійкових вузлів. В даному випадку складається з набору видимих вузлів $v \in \{0,1\}^D$ та ряду латентних вузлів $h^{(1)} \in \{0,1\}^{F1}$, $h^{(2)} \in \{0,1\}^{F2}$, ..., $h^{(L)} \in \{0,1\}^{FL}$. Зав'язків між однорівневими вузлами відсутні. Ймовірність глибинності машини Больцмана розраховується за формулою.

$$p(v) = \frac{1}{Z} \sum_{h} e^{\sum_{ij} W_{ij}^{(1)} v_i h_j^{(1)} + \sum_{jl} W_{jl}^{(2)} h_j^{(1)} h_l^{(2)} + \sum_{lm} W_{lm}^{(3)} h_l^{(2)} h_m^{(3)}}.$$

Глибинній мережі переконань верхні два шари створюють обмежену машину Больцмана, ті що знизу шари створюють орієнтовану створюючу модель. Зв'язки є симетричними, так як вона є графовою моделлю, ймовірнісна модель, для якої умовні залежності між випадковими змінними виражено графом, слої що знаходяться на знизу створюють орієнтовану створюючу модель.

Що глибинні мережі переконань так глибинна машина Больцмана, вони можуть отримати навчання складних та абстрактних внутрішніх представлень входу в ситуаціях при використанні розпізнавання об'єктів та мовлення, застосовуючи обмежені мічені дані для настроювання тонкого представлення, зробленого з великого представлення немічених сенсорних даних на вхід [2].

Швидкість роботи глибинної машини Больцмана, зменшує її продуктивність та функціональність. Для даної методики є непіддатливим навчання максимальної точної правдоподібності, тому виконується навчання наближеної максимальної правдоподібності. Також можна використати осереднене поле для

оцінювання залежних від даних очікувань, наближувано очікуваної достатньої моделі Монте-Карло марковських ланцюгів (рис. 2), це клас алгоритмів для вибірки з розподілу ймовірностей на базі побудови такого ланцюга Маркова, що має бажаний розподіл як свій рівноважний розподіл

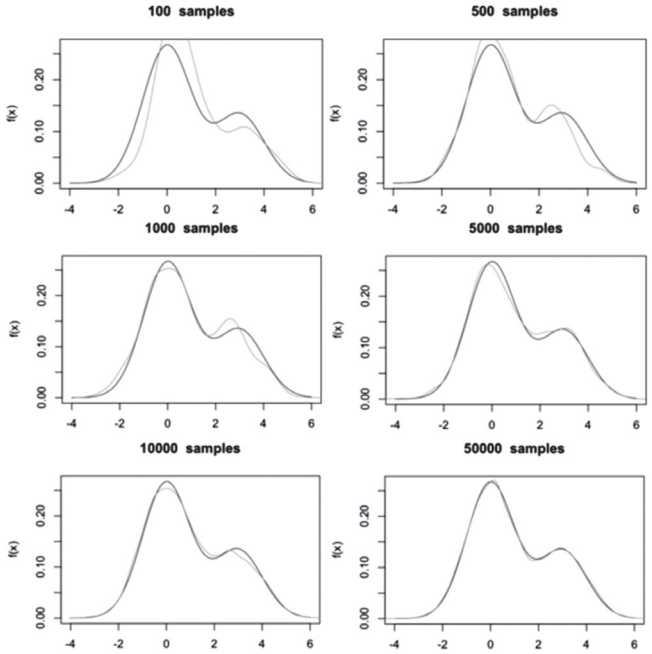


Рис. 2. Приклад Монте-Карло марковських ланцюгів

Воно є наближеними, що мусить бути здійснено для кожного перевіреного входу, є у діапазоні 25-50 разів є повільним за єдиним висхідним проходом у глибинній машині переконань. Що робить спільну оптимізацію дуже напрактчною у використанні для великих наборів даних, що в свою чергу дуже сильно обмежує застосування глибинної машини Больцмана у ситуаціях представлення ознак.

Складені автокодувальники, кодувальник – детермінське відображення f_o , перетворює вектор що на вході x на приховане представлення y, O = {W, b}, W є ваговою атрицею, а b є вектором зсуву.

Декодувальник – повертає назад приховане представлення у на відтворений вхід z через g0 [6].

Головна ціль авдокодувальника продиктована поняттям «доброго представлення». Як прикладом можна розглянути класифікатор добре представлення може бути признане, як ефективний класифікатор.

Складені знешумлені автокодувальники частково спотворений вихід очищується. Дану систему було розвернуто в 2010 році Венсаном, разом з особливим підходом до доброго представлення, добре представлення є таким, що може бути надійно отримано зі спотвореного входу, і буде корисним для відновлення відповідного чистого входу [3]. Неявними в цьому визначенні є наступні ідеї:

– представлення вищого рівня є відносно стабільними й стійкими до спотворень входу;

– необхідно виділяти ознаки, що є корисними для представлення розподілу входу.

Даний підхід складається з декількох кроків, спочатку імовірне відображення, що є спотворювальним кроком, потім автокодування та відображається на приховане представлення, останнім кроком виконується мінімізація.

Щоб отримати глибинну архітектуру автокодування, їх накладають один на одного. Після знешумлення та навчання першої функції, можливо натренувати наступний рівень.

Після того як автокодувальник натреновано, його вихід може бути використано як «кероване навчання», (логістична регресія або метод опорних векторів).

Змішані ієрархічно-глибинні моделі компонують глибинні мережі з непараметричних баєсових моделей. Архітектуру які можна використати для навчання є глибинні машини баєса, глибинні автокодувальника та згорткові варіанти. Що забезпечує краще представлення, та швидке навчання, точну класифікацію із даними високої розмірності. Ці архітектури є слабкими в навчанні нових класів на декількох прикладах, так-як всі вузли мережі додані до представлення входу, і повинні бути скориговані разом [5].

Обмеження ступеню свободи знижує кількість параметрів для навчання, допомагаючи навчанню нових класів з кількох прикладів. Ієрархічні баєсові моделі забезпечують навчання з кількох прикладів, наприклад, для комп'ютерного бачення, статистики та когнітивної науки.

В своїй основній ідеї змішані ІГ-архітектури компонують характеристики глибинних мереж та ієрахії баєса. Всі рівні отримують навчання разом, використовуючи максимізації функції внеску, що показує наскільки чутливі функції правдоподібності, залежить від свого параметру.

3. Аналіз та порівняння

Є кілька способів, якими характеризувалася галузь глибинного навчання. Наприклад, 1986 року Ріна Дехтер ввела поняття глибинного навчання першого порядку та глибинного навчання другого порядку в контексті задоволення обмежень. Пізніше глибинне навчання було охарактеризовано як клас алгоритмів машинного навчання, які використовують каскад багатьох шарів вузлів нелінійної обробки для виділяння ознак та перетворення. Кожен наступний шар використовує вихід із попереднього шару як вхід. Алгоритми можуть бути з керованим або спонтанним навчанням, а застосування включають розпізнавання образів та класифікацію.

Вони ґрунтуються на навчанні декількох шарів ознак або представлень даних. Ознаки вищих рівнів виводяться з ознак нижчих рівнів для формування ієрархічного представлення.

Глибинне навчання є частиною ширшої області машинного навчання з навчання представлень даних навчаються кільком рівням представлень, що відповідають різним рівням абстракції; ці рівні формують ієрархію понять.

Ці визначення мають спільними декілька шарів вузлів нелінійної обробки та кероване або спонтанне навчання представлень ознак у кожному з шарів, з формуванням шарами ієрархії від низькорівневих до високорівневих ознак. Побудова шару вузлів нелінійної обробки, що застосовується в алгоритмі глибинного навчання, залежить від розв'язуваної задачі.

Шари, що застосовувалися в глибинному навчанні, включають приховані шари штучної нейронної мережі та набори складних висловлень. Вони також можуть включати латентні змінні, організовані по шарах у глибинних породжувальних моделях, такі як вузли в глибинних мережах переконань та глибинних машинах Больцмана.

Алгоритми глибинного навчання перетворюють свої входи крізь більшу кількість шарів, ніж алгоритми поверхневого навчання. На кожному шарі сигнал перетворюється блоком обробки, таким як штучний нейрон, параметри якого «навчаються» шляхом тренування. Ланцюг перетворень від входу до виходу є шляхом передачі довіри.

Описуючи потенційно причинні зв'язки між входом та виходом, і можуть мати змінну довжину. Для нейронної мережі прямого поширення довжина шляхів передачі довіри, і відтак глибина цієї мережі, є числом прихованих шарів плюс один. Для рекурентних нейронних мереж, в яких сигнал може поширюватися через якийсь шар більше одного разу, має потенційно необмежену довжину. Універсально узгодженого порогу глибини, що відділяв би поверхневе навчання від глибинного, не існує, але більшість дослідників у цій галузі погоджуються, що глибинне навчання має декілька нелінійних шарів (ШПД > 2), а Шмідгубер розглядає ШПД > 10 як дуже глибинне навчання.

Висновок

Розглядаючи різні методи та підходи глибинного навчання, можна зробити висновок в кожному з них є свої переваги та недоліки, це може бути можливість праці над великими об'ємами даних, та бути дуже повільними, або навпаки бути в роботі дуже швидкими в навчанні, але працювати з невеликими об'ємами даними.

В наш час коли передача природної мови є дуже швидким, основним методом якого є Інтернет, це може бути спілкування за допомогою додатків які забезпечують передачу мови від одної особи до іншої або групи осіб. Спілкування за допомогою набору коротких текстових повідомлень від однієї особи до іншої або групи людей.

Також може бути прослуховування або ознайомлення інформації на різних інтернет ресурсах.

Тому маючи велику кількість оточеної нас природної мови, потрібні інструменти у вигляди натренованих нейронних мереж, що можуть забезпечити обробку мови на виявлення неправильної поведінки однієї особи до іншої, харасмент, що може відбиватися на підставі сексуального характеру, расового та інших ознак.

Можливий один із варіантів розшифрування закодованої мови, або ж можливо навпаки закодувати, можливо використати метод кодування декодування мережі, де можливо навчити систему та використовувати дані інструменти для роботи, в тих чи інших цілях.

Опрацювання природної мови, може також нести вивчення тієї чи іншої ситуації для отримання інформації, для цього можливо застосувати один із методів глибинного навчання, за допомогою якого можна провести навчити нейронної мережі та застосувати для вирішення.

Моделі рекурсивного глибокого навчання можуть вирішувати багато мовних задач, що включають в себе передбачення на рівні слова та речення як безперервного, так і дискретного характеру. Одним із безлічі існуючих рішень для вирішення цієї задачі є використання рекурсивних або рекурентних методів для розрахунку подань рівня абзацу або документа.

Іншою реалізацією роботи для глибиних моделей – це логічні міркування першого порядку, які можуть бути використані для отримання правильної інформації з баз знань за допомогою питань природної мови. Переглянуті моделі в цій статті можуть бути розширені, щоб спільно моделювати мову, образи та бази знань в одній цілісній семантичній структурі.

Список літератури:

[1] *J. Le.* (2018). The 7 NLP Techniques That Will Change How You Communicate in the Future (Part I) | [Online]. Available: https://heartbeat.fritz.ai/the-7-nlp-techniques-that-will-change-how-you- communicate-in-the-future-part-i-f0114b2f0497

[2] *M. Bates* (1995). Models of natural language understanding | [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC40721/

[3] *R. Socher*, Recursive deep learning for natural language processing and computer vision | Stanford University, 2014, pp. 8-120.

[4] *Afanasieva I.* Data exchange model in the Internet of Things concept / I. Afanasieva, N. Golian, O. Hnatenko, Y. Daniiel, K. Onyshchenko // Telecommunications and Radio Engineering, New York, 2019. – 10(78). – p. 869-878

[5] *Onyshchenko A.* Adaptive method of training neural networks / A. Onyshchenko, K. Onyshchenko // Technique and technology. Science, Research, Development #29. Gdansk, 2020. – p. 9-11.

[6] Web-site - https://en.wikipedia.org/wiki/Deep_learning/

Надійшла до редколегії 28.10.2021

УДК 519.62

DOi 10.30837/ bi.2021.2(97).06

І.В. Кириченко[1], В.Д. Рошка[2]

[1] кандидат технічних наук, старший викладач, кафедра програмної інженерії, Харківський Національний Університет Радіоелектроніки, Харків, Україна, iryna.kyrychenko@nure.ua, ORCID iD: 0000-0002-7686-6439

[2] студент бакалаврату, кафедра програмної інженерії, Харківський Національний Університет Радіоелектроніки, Харків, Україна, veronika.roshka@nure.ua, ORCID iD: 0000-0002-1704-9519

ПОРІВНЯННЯ ЕФЕКТИВНОСТІ АЛГОРИТМІВ ПОШУКУ ШЛЯХУ ПРИ РОЗРОБЦІ ІГРОВОГО ШТУЧНОГО ІНТЕЛЕКТУ

З точки зору ігор справжній ШІ далеко виходить за рамки вимог розважального програмного проекту. В іграх така міць не потрібна. Ігровий ШІ не повинен бути наділений почуттями та самосвідомістю, йому немає необхідності вчитися чогось за межами рамок ігрового процесу. Справжня мета ШІ в іграх полягає в імітації розумної поведінки та у наданні гравцеві переконливого, правдоподібного завдання. Щоб штучний інтелект міг приймати осмислені рішення, йому необхідно будь-яким чином сприймати середовище, в якому він знаходиться. У найпростіших системах таке сприйняття може обмежуватися простою перевіркою положення об'єкта гравця. У складних системах потрібно визначати основні показники і якості ігрового світу, наприклад можливі маршрути для пересування, наявність природних укриттів на території, області конфліктів.

При цьому розробникам необхідно вигадувати спосіб виявлення та визначення основних властивостей ігрового світу, важливих для ШІ системи. Наприклад, укриття на місцевості можуть бути визначені дизайнерами рівнів або заздалегідь обчислені при завантаженні або компіляції карти рівня. Деякі елементи необхідно обчислювати на льоту, наприклад, карти конфліктів і найближчі загрози.

Пошук шляху або pathfinder — визначення комп'ютером найоптимальнішого маршруту між двома точками. З ним часто можна зіткнутися при розробці ігор, наприклад, якщо в них є вороги, що вільно пересуваються. Потрібно не просто прагнемо знайти найкоротшу відстань, а також необхідно враховувати і тривалість руху. Для пошуку цього шляху можна використовувати алгоритм пошуку за графом, який застосовується, якщо карта є графом. A* часто використовується як алгоритм пошуку за графом. Пошук в ширину — це найпростіший алгоритм пошуку за графом, тому в цій статті почнемо розбиратися з нього і поступово перейдемо до A*.

Карти, на яких шляхи прокладаються за допомогою алгоритму «Зіткнутися і повернути», дозволяють пристосуватися до карт, що змінюються. Але у стратегічних іграх гравці не можуть чекати, поки їхні війська розберуться з прокладанням маршрутів. Крім того, карти шляхів можуть бути дуже великими, і на вибір правильного шляху на таких картах витрачатиметься дуже багато ресурсів. У таких ситуаціях на допомогу приходить алгоритм пошуку шляхів.

ІГРОВИЙ ШТУЧНИЙ ІНТЕЛЕКТ, ПОШУК ШЛЯХУ, ПОШУК В ШИРИНУ, АЛГОРИТМ ДЕЙКСТРИ, ЕВРИСТИЧНИЙ АЛГОРИТМ, ЖАДКОВИЙ ПОШУК, АЛГОРИТМ A*

Iryna Kyrychenko, Veronika Roshka. Comparison of the efficiency of path finding algorithms in the development of game artificial intelligence. In terms of games, true AI goes far beyond the requirements of an entertainment software project. In games, this power is not needed. Game AI does not have to be endowed with feelings and self-awareness; it does not need to learn anything outside of the gameplay. The real purpose of AI in games is to mimic intelligent behavior and to present the player with a compelling, believable challenge. In order for artificial intelligence to make meaningful decisions, it needs to somehow perceive the environment in which it is located. In simple systems, this perception can be limited to simply checking the position of the player's object. In more complex systems, it is required to determine the main characteristics and properties of the game world, for example, possible routes for movement, the presence of natural shelters on the ground, and areas of conflict.

At the same time, developers need to come up with a way to identify and define the main properties of the game world that are important for the AI system. For example, terrain cover can be predefined by the level designers or calculated in advance when loading or compiling a level map. Some elements need to be calculated on the fly, such as conflict maps and nearby threats.

Pathfinder – the computer determines the best route between two points. You can often encounter it when developing games, for example, if there are free-moving enemies in them. It is necessary not only to find the shortest distance, but also to take into account the duration of the movement. To find this path, you can use the graph search algorithm, which is applicable if the map is a graph. A * is often used as a graph search algorithm. Breadth First Search is the simplest graph search algorithm, so in this article we will start to understand it and gradually move on to A*.

Bump-and-turn maps allow you to adapt to changing maps. But in strategy games, players can't wait for their troops to figure out the routes. In addition, path maps can be very large, and choosing the right path on such maps will consume a lot of resources. In such situations, the pathfinding algorithm comes to the rescue.

GAME ARTIFICIAL INTELLIGENCE, WAY SEARCH, BREADTH FIRST SEARCH, DIJKSTRA'S ALGORITHM, HEURISTIC ALGORITHM, GREEDY SEARCH, A * ALGORITHM

Кириченко И. В., Рошка В.Д. Сравнение эффективности алгоритмов поиска пути при разработке игрового искусственного интеллекта. С точки зрения игр подлинный ИИ далеко выходит за рамки требований развлекательного программного проекта. В играх такая мощь не нужна. Игровой ИИ не должен быть наделен чувствами и самосознанием, ему нет необходимости обучаться чему-либо за пределами рамок игрового процесса. Подлинная цель ИИ в играх состоит в имитации разумного поведения и в предоставлении игроку убедительной, правдоподобной задачи. Чтобы искусственный интеллект мог принимать осмысленные решения, ему необходимо каким-либо образом воспринимать среду, в которой он находится. В простых системах такое восприятие может ограничиваться простой проверкой положения объекта игрока. В более сложных системах требуется определять основные характеристики и свойства игрового мира, например возможные маршруты для передвижения, наличие естественных укрытий на местности, области конфликтов.

При этом разработчикам необходимо придумывать способ выявления и определения основных свойств игрового мира, важных для системы ИИ. Например, укрытия на местности могут быть заранее определены дизайнерами уровней или заранее вычислены при загрузке или компиляции карты уровня. Некоторые элементы необходимо вычислять на лету, например карты конфликтов и ближайшие угрозы.

Поиск пути или pathfinder — определение компьютером самого оптимального маршрута между двумя точками. С ним часто можно столкнуться при разработке игр, например, если в них есть свободно передвигающиеся враги. Необходимо не просто найти кратчайшее расстояние, также нужно учесть и длительность движения. Для поиска этого пути можно использовать алгоритм поиска по графу, который применим, если карта представляет собой граф. A* часто используется в качестве алгоритма поиска по графу. Поиск в ширину — это простейший из алгоритмов поиска по графу, поэтому в этой статье начнем разбираться с него и постепенно перейдём к A*.

Карты, на которых пути прокладываются с помощью алгоритма «Столкнуться и повернуть», позволяют приспособиться к изменяющимся картам. Но в стратегических играх игроки не могут ждать, пока их войска разберутся с прокладыванием маршрутов. Кроме того, карты путей могут быть очень большими, и на выбор правильного пути на таких картах будет расходоваться очень много ресурсов. В таких ситуациях на помощь приходит алгоритм поиска путей.

ИГРОВОЙ ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ, ПОИСК ПУТИ, ПОИСК В ШИРИНУ, АЛГОРИТМ ДЕЙКСТРЫ, ЭВРИСТИЧЕСКИЙ АЛГОРИТМ, ЖАДНЫЙ ПОИСК, АЛГОРИТМ A*

Вступ

Комп'ютерним NPC не можна просто сказати: «Гей ти, іди до вежі!», як людям. Їм потрібно прописати точний набір команд (іди один метр вліво, потім метр вперед і т.д.) або вказати шлях з проміжних точок, щоб вони успішно дісталися точки А до точки В. Саме знаходженням цих проміжних точок і займається алгоритм пошуку шляху. Але цим завдання не обмежуються. Додатково він може визначати відстань до мети і те, чи взагалі можливо дістатися до якоїсь точки.

Є такий вид ігор-головоломок – лабіринт Мінотавра. Гравця вміщують на сітчасте поле зі стінками. Одна із клітин поля є виходом, а на іншій стоїть монстр. Завдання гравця – дістатися до виходу раніше, ніж монстр добереться до нього. Гравець і монстр ходять по черзі на скільки клітин, але монстр завжди переміщується швидше гравця.

Гравець би постійно програвав, але монстр має мінус: він хоч і швидкий, але тупий. Він завжди намагається дістатися гравця найкоротшим способом і не звертає уваги на те, що в лабіринті є стіни. Якщо монстру на заваді зустрічається стіна, він просто стоїть і сумно зітхає перед нею, не намагаючись обійти.

У таких іграх дурість глупість виправдана, оскільки вона є частиною геймплей головоломки. В інших іграх користувач очікує, що ворог почне шукати шляхи обходу, а не застрягатиме в камінні та деревах.

Завдання пошуку шляху складається з двох етапів: адаптування ігрового світу до математичної моделі та пошук у цій моделі шляху між двома точками.

1. Адаптування ігрового світу

Комп'ютер не може просто подивитися на камінь та сказати, що це камінь. Тому йому треба описати ігровий світ у вигляді чисел і вибрати набір ознак, якими буде визначатися, що між двома точками можна пройти.

Більшість алгоритмів будується на графах. Тому шляхи проходу ігровим світом перетворюють на графи. Як точки-кружочки зазвичай беруть мінімальну площу світу, в якій пересування від одного краю до другого відбувається миттєво (або за досить маленький час, щоб вважати його миттєвим). Інакше висловлюючись, у іграх ці кружечки — це позиції персонажа. Ребра ж позначають той факт, що персонаж із першого шматка світу може перейти на другий. Приклад зображено на рисунку 1.

Одна з найпростіших варіацій графа — ігрова сітка, де кружальцями є клітини, а ребра проводяться між будь-якими сусідніми клітинами, вільними від перешкод.

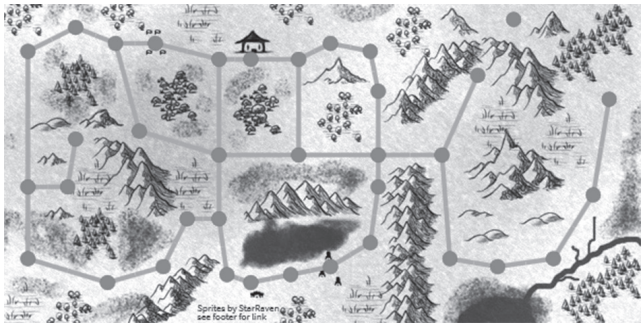


Рис. 1. Приклад побудови графа

Найпростіше розібратися в цьому на прикладі покрокових ігор, у яких сітку видно неозброєним оком, як Sneaky-Sneaky або Return of the Necrodancer (див. рис. 2).

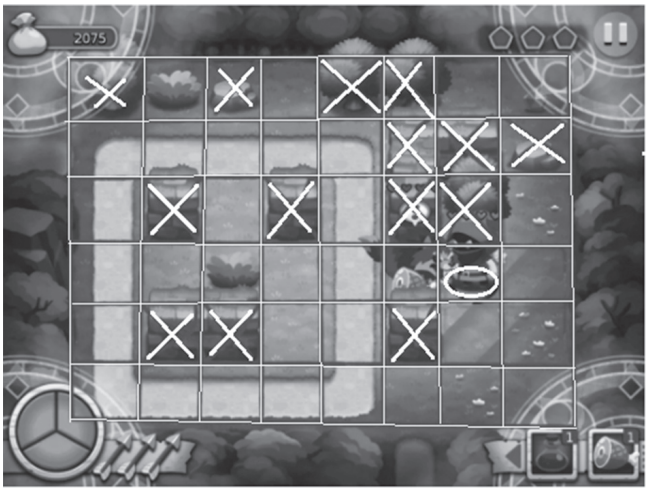


Рис. 2. Зразкова сітка у грі Sneaky-Sneaky

Клітини, на яких стоять непереборні перешкоди, позначаються хрестиком, а місця, що проходять – порожнечею. Або якщо говорити мовою програмування, клітини позначаються 0 або 1, де 0 означає, що пройти не можна, а 1 – що пройти можна. Більш складні ігри на кшталт Heroes of Might and Magic III відрізняються лише тим, що їхня сітка малюється більш довільно, а клітини, які розташовані на різних типах землі, забирають різну кількість кроків.

При роботі з тривимірними світами можна використовувати двомірну карту проекції ландшафту із зазначенням висот у точках. І тут кожному осередку написана її висота, а прохідність визначається через різницю висот сусідніх клітин.

Найчастіше адаптація ігрового світу – це найскладніша частина, тому що алгоритми пошуку оптимальних маршрутів винайдено давно, можна навіть знайти готовий код для них. Тому розробники впроваджують один з алгоритмів і потім експериментують з різними моделями та варіаціями порівнянь двох шляхів, або комбінують різні алгоритми у спробах досягти найкращого та найменш витратного результату.

2. Алгоритми пошуку шляху

Як би людина шукала вихід, якби її раптово висадили посеред лабіринту? Наприклад, він міг би скористатися правилом «однієї руки», відзначати обстеженні шляхи мотузкою або крейдою, доки не знайде вихід. Більшість алгоритмів діють за схожою схемою, за винятком того, що комп'ютер може розмножуватися та «обмацувати» одночасно кілька шляхів. Розрізняються алгоритми точністю і швидкістю отримання результату.

Пошук у ширину виконує дослідження поступово в усіх напрямках. Приклад дивись на рис. 3*а*. Це неймовірно корисний алгоритм, як для звичайного пошуку шляху, але й процедурної генерації карт, пошуку шляхів течії, карт відстаней та інших типів аналізу карт.

Алгоритм Дейкстри (також званий пошуком із рівномірною вартістю) дозволяє нам ставити пріоритети дослідження шляхів. Замість рівномірного дослідження всіх можливих шляхів він віддає перевагу шляхам із низькою вартістю. Приклад дивись на рис. 3*б*. Ми можемо задати зменшені витрати, щоб алгоритм рухався дорогами, підвищену вартість, щоб уникати лісів та ворогів, та багато іншого. Коли вартість руху може бути різною, ми використовуємо його замість пошуку у ширину.

A* – це модифікація алгоритму Дейкстри, оптимізована для єдиної кінцевої точки. Алгоритм Дейкстри може знаходити шляхи до всіх точок, A * знаходить шлях до однієї точки. Він віддає пріоритет шляхам, що ведуть ближче до мети (див. рис. 3*в*).

а *б* *в*

Рис. 3 (*а*, *б*, *в*). Графічне представлення роботи алгоритмів

Почнемо з найпростішого — пошуку у ширину, і будемо додавати функції, поступово перетворюючи його на A*.

3. Пошук у ширину

Ключова ідея всіх цих алгоритмів полягає в тому, що ми відстежуємо стан кільця, що розширюється, яке називається кордоном. У сітці цей процес іноді називається заливкою (flood fill), але та сама техніка застосовна і для карт без сіток.

Щоб це реалізувати, необхідно повторити ці кроки, поки кордон не стане порожнім:

1) Вибираємо та видаляємо крапку з кордону.

2) Позначаємо точку як відвідану, щоб знати, що не потрібно обробляти її повторно.

3) Розширюємо кордон, дивлячись на сусідів. Усіх сусідів, яких ми ще не бачили, додаємо до кордону.

Алгоритм описується лише в десяти рядках коду на Python:

```
frontier = Queue()
frontier.put(start )
visited = {}
visited[start] = True

while not frontier.empty():
    current = frontier.get()
    for next in graph.neighbors(current):
        if next not in visited:
            frontier.put(next)
            visited[next] = True
```

У цьому вся циклі полягає вся сутність алгоритмів пошуку за графом цієї статті, зокрема і A*. Але як знайти найкоротший шлях? Цикл насправді не створює шляхів, він просто каже нам, як відвідати всі точки на карті. Так вийшло тому, що пошук у ширину можна використовувати для набагато більшого, ніж просто пошук шляхів. У цій статті показується, як він застосовується в іграх tower defense, але його також можна використовувати в картах відстаней, у процедурній генерації карт та багато іншого. Однак тут ми хочемо використовувати його для пошуку шляхів, тому давайте змінимо цикл так, щоб відстежувати, звідки ми прийшли для кожної відвіданої точки, і перейменуємо visited в came_from:

```
frontier = Queue()
frontier.put(start )
came_from = {}
came_from[start] = None

while not frontier.empty():
    current = frontier.get()
    for next in graph.neighbors(current):
        if next not in came_from:
            frontier.put(next)
            came_from[next] = current
```

Тепер came_from для кожної точки вказує місце, з якого ми прийшли. Нам цього достатньо, щоб відтворити цілий шлях. Подивіться, як стрілки показують зворотний шлях до початкової позиції на рис. 4.

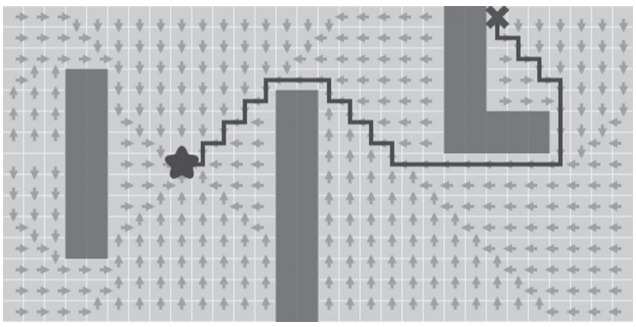


Рис. 4. Приклад реалізації

Код відтворення шляхів простий: слідуємо за стрілками назад від мети на початок. Шлях – це послідовність ребер, але іноді простіше зберігати лише вузли:

```
current = goal
path = [current]
while current != start:
    current = came_from[current]
    path.append(current)
path.append(start) # optional
path.reverse() # optional
```

Такий найпростіший алгоритм пошуку шляхів. Він працює не тільки в сітках, як показано вище, а й у будь-якій структурі графів. У підземеллі точки графа можуть бути кімнатами, а ребра – дверима між ними. У платформері вузли графа можуть бути локаціями, а ребра – можливими діями: переміститися вліво, вправо, підстрибнути, стрибнути вниз. У цілому нині можна сприймати граф як і дії, що змінюють стан.

Пошук у ширину на кожному N циклі знаходить всі точки, яких можна дійти за N кроків. Щоб отримати шлях, потрібно запам'ятовувати як перевірені точки, а ще й послідовність точок до них. Або хоча б попередню точку, з якої ми прийшли на цю, щоб кроками відновити маршрут.

Пошук у ширину добре працює, якщо перехід між сусідніми точками завжди займає однакову кількість часу (або іншого параметра, яким ми намагаємося мінімізувати шлях). Але якщо це не так, то він перетворюється на алгоритм Дейкстри.

4. Ранній вихід

Ми знайшли шляхи з однієї точки до всіх інших точок. Часто нам не потрібні всі шляхи, нам просто потрібен шлях між двома точками. Ми можемо припинити розширювати кордон, як тільки знайдемо нашу мету. Подивіться, як межа перестає розширюватись після знаходження мети (див. рис. 5).

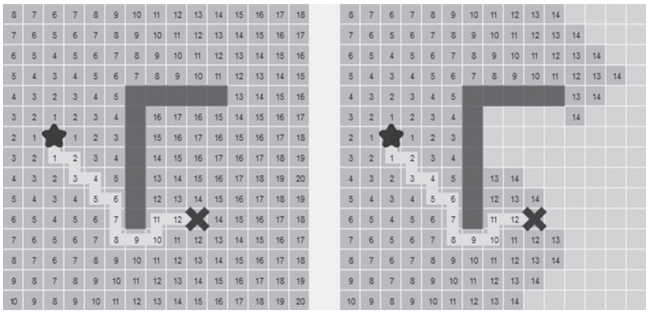


Рис. 5. Перевага раннього виходу

Код досить прямолінійний:

```
frontier = Queue()
frontier.put(start )
came_from = {}
came_from[start] = None
while not frontier.empty():
    current = frontier.get()
    if current == goal:
        break
    for next in graph.neighbors(current):
        if next not in came_from:
            frontier.put(next)
            came_from[next] = current
```

5. Вартість переміщення. Алгоритм Дейкстри

Алгоритм Дейкстри працює з моделями, де відстані між точками можуть бути різними. Наприклад, у Heroes of Might and Magic III прохід по снігу витрачає на 50% більше очок переміщення, так що умовно можна порахувати, що прохід по снігу займає 1.5 кроку замість 1 по звичайній землі. У цьому випадку обхід снігу може бути швидшим, ніж прохід безпосередньо по ньому, хоч візуально і буде пройдено більшу відстань.

Ще одним прикладом є діагональний рух у сітці, який коштує більше, ніж рух по осях. Нам потрібно, щоби пошук шляху враховував цю вартість. Давайте порівняємо кількість кроків від початку (ліворуч) з відстанню від початку (справа), зображених на рис. 6.

Рис. 6. Порівняння кількості кроків із відстанню

Для цього нам потрібен алгоритм Дейкстри (також званий пошуком із рівномірною вартістю). Від пошуку у ширину алгоритм Дейкстри відрізняє пара моментів. Окрім збереження вже перевірених точок, зберігається ще й кількість кроків, витрачених на те, щоби дістатися до них. Крапки виключаються з подальшої перевірки не тоді, коли були вже перевірені, а лише у випадку, якщо попередній знайдений шлях до неї займав меншу кількість кроків. Крапки у списку розглядаються не по порядку, спочатку вибираються ті, до яких менше йти. Нам потрібно відслідковувати вартість руху, тому додаємо нову змінну cost_so_far, щоб слідкувати за загальною вартістю руху початкової точки. Оцінюючи точок нам потрібно враховувати вартість пересування. Давайте перетворимо нашу чергу на чергу з пріоритетами. Менш очевидно те, що у нас може вийти так, що одна точка відвідується кілька разів із різною вартістю, тому потрібно трохи поміняти логіку. Замість додавання точки до кордону у випадку, коли точку жодного разу не відвідували, ми додаємо її, якщо новий шлях до точки кращий, ніж найкращий попередній шлях.

```
  frontier = PriorityQueue()
  frontier.put(start, 0)
  came_from = {}
  cost_so_far = {}
  came_from[start] = None
  cost_so_far[start] = 0

  while not frontier.empty():
    current = frontier.get()
    if current == goal:
      break
    for next in graph.neighbors(current):
      new_cost = cost_so_far[current] +
graph.cost(current, next)
      if next not in cost_so_far or new_cost <
cost_so_far[next]:
        cost_so_far[next] = new_cost
        priority = new_cost
        frontier.put(next, priority)
        came_from[next] = current
```

Використання черги з пріоритетами замість звичайної черги змінює спосіб розширення кордону. Контурні лінії дозволяють це побачити.На рисунку 7 представлені контури, ліворуч – пошук у ширину, праворуч – алгоритм Дейкстри. Кордон розширюється повільніше через ліси, і пошук найкоротшого шляху виконується навколо центрального лісу, а не через нього:

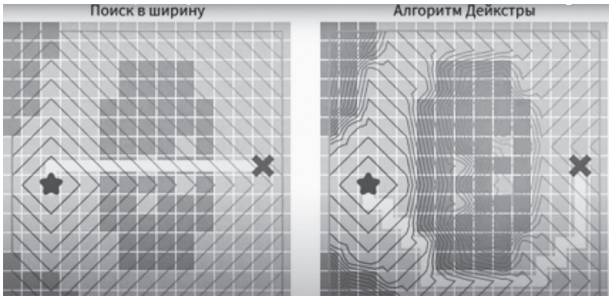


Рис. 7. Розширення кордону

Вартість руху, що відрізняється від 1, дозволяє нам досліджувати цікавіші графи, а не лише сітки.

6. Евристичний пошук

У пошуку в ширину та алгоритмі Дейкстри кордон розширюється у всіх напрямках. Це логічний вибір, якщо ви шукаєте шлях до всіх точок або до безлічі точок. Однак, зазвичай пошук виконується тільки для однієї точки. Давайте зробимо так, щоб межа розширювалася до мети більше, ніж у інших напрямках. По-перше, визначимо евристичну функцію, яка повідомляє нам, наскільки ми близькі до мети:

```
def heuristic(a, b):
  # Manhattan distance on a square grid
  return abs(a.x - b.x) + abs(a.y - b.y)
```

В алгоритмі Дейкстри порядку черги з пріоритетами ми використовували відстань від початку. У жадібному пошуку за першим найкращим збігом для порядку черги з пріоритетами ми замість цього використовуємо оцінену відстань до мети. Крапка, найближча до мети, буде вивчена першою. У коді використовується черга з пріоритетами пошуку в ширину, але не застосовується cost_so_far з алгоритму Дейкстри:

```
frontier = PriorityQueue()
frontier.put(start, 0)
came_from = {}
came_from[start] = None

while not frontier.empty():
  current = frontier.get()

  if current == goal:
    break

  for next in graph.neighbors(current):
    if next not in came_from:
      priority = heuristic(goal, next)
      frontier.put(next, priority)
      came_from[next] = current
```

Результат виконання представлений на рис. 8, 9.

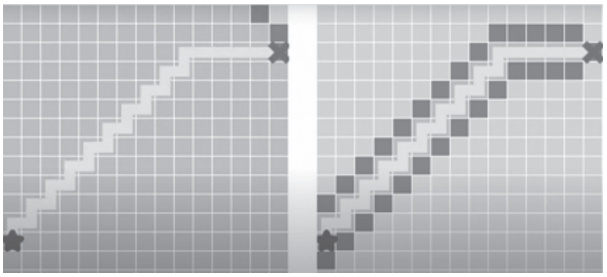


Рис. 8. Результат виконання коду

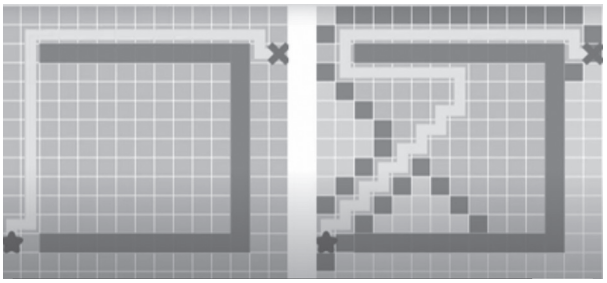


Рис. 9. Результат на більш складній карті

Ці шляхи не є найкоротшими. Отже, цей алгоритм працює швидше, коли перешкод не дуже багато, але шляхи не надто оптимальні. Чи можна це виправити? Звичайно.

7. Алгоритм A*

Алгоритм Дейкстри гарний у пошуку найкоротшого шляху, але він витрачає час на дослідження всіх напрямків, навіть безперспективних. Жадібний пошук досліджує перспективні напрями, але може знайти найкоротший шлях. Алгоритм A* використовує і справжнє відстань від початку, і оцінене відстань до мети.

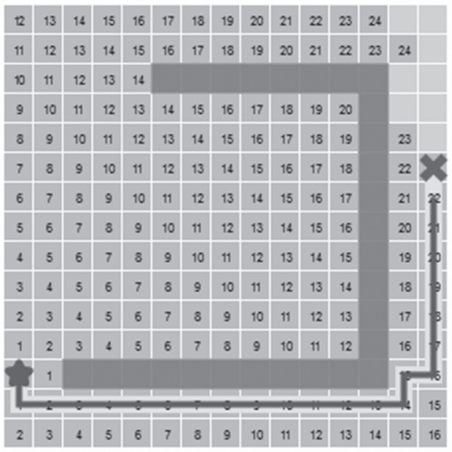
Код дуже схожий на алгоритм Дейкстри:

```
frontier = PriorityQueue()
frontier.put(start, 0)
came_from = {}
cost_so_far = {}
came_from[start] = None
cost_so_far[start] = 0

while not frontier.empty():
  current = frontier.get()

  if current == goal:
    break

  for next in graph.neighbors(current):
    new_cost = cost_so_far[current] + graph.cost(current, next)
    if next not in cost_so_far or new_cost < cost_so_far[next]:
      cost_so_far[next] = new_cost
      priority = new_cost + heuristic(goal, next)
      frontier.put(next, priority)
      came_from[next] = current
```

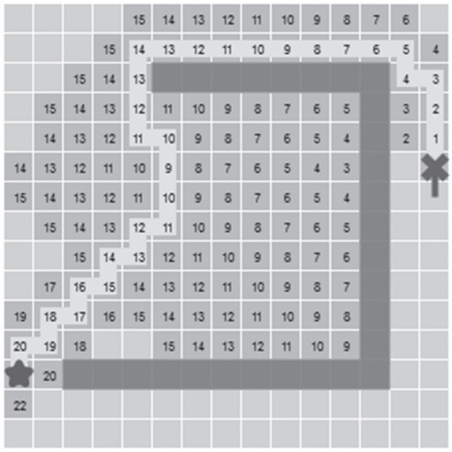
Порівняємо алгоритми. Алгоритм Дейкстри обчислює відстань від початкової точки (див. рис. 10а). Жадібний пошук за першим найкращим збігом оцінює відстань до точки мети (див. рис. 10*б*). A* використовує суму цих двох відстаней(див. рис. 10*в*).

Як видно з ілюстрацій, жадібний пошук знаходить правильну відповідь, A* теж його знаходить, досліджуючи ту саму область. Коли жадібний пошук по першому найкращому знаходить неправильну відповідь (довший шлях), A* знаходить правильний, як і алгоритм Дейкстри, але все одно досліджує менше, ніж алгоритм Дейкстри.

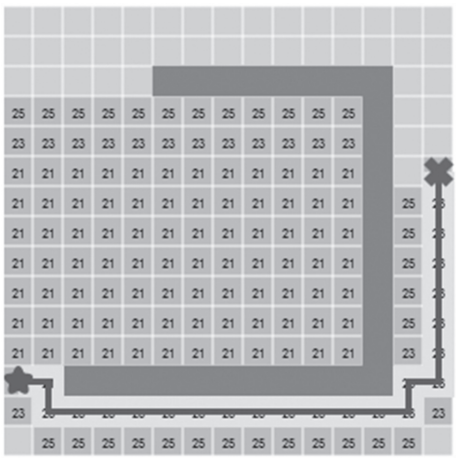
A* бере найкраще від двох алгоритмів. Оскільки евристика не оцінює відстані повторно, A* не використовує евристику для пошуку відповіді. Він знаходить оптимальний шлях, як алгоритм Дейкстри.



а) Алгоритм Дейкстри



б) Жадібний пошук



в) A* пошук

Рис. 10 (а, б, в) – Порівняння алгоритмів

A* використовує евристику для зміни порядку вузлів, щоб підвищити ймовірність раннього знаходження вузла мети.

Алгоритм A* намагається всидіти на обох стільцях одразу: знайти найкоротшу відстань, але зробити це за менший час, ніж алгоритм Дейкстри. Тому наступна точка на розгляд вибирається за мінімальною сумою відстаней до початку та до кінця шляху. Це єдина відмінність від попередніх алгоритмів.

Висновки

З вищесказаного виникає питання. Який алгоритм варто використовувати для пошуку шляхів на карті?

Якщо вам потрібно знайти шляхи з або до всіх точок, використовуйте пошук у ширину або алгоритму Дейкстри. Використовуйте пошук у ширину, якщо вартість руху однакова. Використовуйте алгоритм Дейкстри, якщо вартість руху змінюється.

Якщо потрібно знайти шляхи до однієї точки, використовуйте жадібний пошук за найкращим першим або A *. Найчастіше варто віддати перевагу A*. Коли є спокуса використовувати жадібний пошук, подумайте над застосуванням A* з «неприпустимою» евристикою.

А що щодо оптимальних шляхів? Пошук у ширину та алгоритм Дейкстри гарантовано знайдуть найкоротший шлях по графу. Жадібний пошук не обов'язково його знайде. A* гарантовано знайде найкоротший шлях, якщо евристика ніколи не більша за справжню відстань. Коли евристика стає меншою, A* перетворюється на алгоритм Дейкстри. Коли евристика стає більшою, A* перетворюється на жадібний пошук за найкращим першим збігом.

А як щодо продуктивності? Найкраще усунути непотрібні точки графа. Якщо ви використовуєте сітку, прочитайте це. Зменшення розміру графа допомагає всім алгоритмам пошуку за графами. Після цього використовуйте найпростіший з можливих алгоритмів. Прості черги виконуються швидше. Жадібний пошук зазвичай виконується швидше, ніж алгоритм Дейкстри, але не забезпечує оптимальних шляхів. Для більшості завдань пошуку шляхів оптимальним вибором є A*.

А що щодо використання не на картах? У статті використовувалися карти тому, що простіше пояснити роботу алгоритму. Однак ці алгоритми пошуку за графами можна використовувати на будь-яких графах, не тільки на ігрових картах, у статті представлений код алгоритму у вигляді, що не залежить від двовимірних сіток. Вартість руху на картах перетворюється на довільні ваги ребер графа. Евристики перенести на довільні карти непросто, необхідно створювати евристику кожного типу графа. Для плоских карток хорошим вибором будуть відстані, тому тут ми використовували їх.

Не забувайте, що пошук за графами – це лише одна частина того, що вам потрібне. Сам по собі A* не обробляє такі аспекти, як спільний рух, переміщення перешкод, зміна карти, оцінка небезпечних областей, формації, радіуси повороту, розміри об'єктів, анімацію, згладжування шляхів та багато іншого.

Є й інші алгоритми, що дають переваги за різних умов. Самі алгоритми пошуку шляху досить прості, але вони працюють для сферичних ідеальних умов у вакуумі. У реальних іграх є багато нюансів, які можуть ускладнити алгоритм. Наприклад, може бути кілька варіантів переміщень, повороти, перешкоди можуть рухатись, можуть бути вузькі місця, в яких ворогам бажано не товпитися. Кожна така умова робить рух цікавішим, але й уповільнює роботу алгоритму. Через це доводиться щось спрощувати або викручуватись по-іншому для оптимізації роботи гри.

Список літератури:

[1] Grid pathfinding optimizations // URL: https://www.redblobgames.com/ pathfinding/grids/algorithms.html (date of access: 11.01.2022).

[2] Grids and Graphs // URL: https://www.redblobgames.com/ pathfinding/ grids/graphs.html (date of access: 11.01.2022).

[3] Введення в алгоритм A* // URL: https://habr.com/ru/post/331192/ (date of access: 11.01.2022).

[4] Пошук шляху, або як вороги в іграх знаходять дорогу // URL: https://dtf.ru/gamedev/709133-poisk-puti-ili-kak-vragi-v-igrah-nahodyat-dorogu (date of access: 11.01.2022).

[5] Map representations // URL: http://theory.stanford.edu/~amitp/Game Programming/MapRepresentations.html (date of access: 11.01.2022).

[6] Algorithm A * Implementation // URL: https://habr.com/ru/post/331220/ (date of access: 11.01.2022).

[7] Admissible heuristic // URL: https://en.wikipedia.org/wiki/Admissible _heuristic (date of access: 11.01.2022).

[8] Створення штучного інтелекту для ігор – від проектування до оптимізації // URL: https://habr.com/ru/company/intel/blog/265679/ (date of access: 11.01.2022).

[9] Як створити ігровий ШІ: гайд для початківців // URL: https://habr.com/ru/company/pixonic/blog/428892/ (date of access: 11.01.2022).

[10] Як створити ігровий ШІ: гайд для початківців // URL: https://habr.com/ru/post/420219/ (date of access: 11.01.2022).

[11] Не зовсім людина: штучний інтелект в іграх // URL: https://skillbox.ru/media/gamedev/iskusstvennyy-intellekt-v-igrakh/ (date of access: 11.01.2022).

[12] Pathfinding Demystified (Part I): Generic Search Algorithm // URL: https://www.gabrielgambetta.com/generic-search.html (date of access: 11.01.2022).

[13] Generalized Platformer AI Pathfinding // URL: https://www.gamedev.net/articles/programming/artificial-intelligence/generalized-platformer-ai-pathfinding-r3924/ (date of access: 11.01.2022).

[14] Алгоритм Дейкстри. Пошук оптимальних маршрутів на графі // URL: https://habr.com/ru/post/111361/ (date of access: 11.01.2022).

Надійшла до редколегії 14.09.2021

УДК 004.89 DOi 10.30837/ bi.2021.2(97).07

С.О. Мар'їн[1], І.О. Побіженко[2], Т.Г. Білова[3], В.М. Дьоміна[4]

[1]кандидат технічних наук, доцент, доцент кафедри програмної інженерії ХНУРЕ,
м. Харків, Україна, serhiy.maryin@nure.ua, ORCID iD: 0000-0002-2012-7477

[2]кандидат технічних наук, доцент, доцент кафедри інформаційних технологій ХДАК,
м. Харків, Україна, irina_pob@ukr.net, ORCID iD: 0000-0002-0723-1878

[3]кандидат технічних наук, доцент, доцент, доцент кафедри системотехніки ХНУРЕ,
м. Харків, Україна, tetiana_bielova@nure.ua, ORCID iD: 0000-0002-1085-7361

[4]кандидат технічних наук, доцент, доцент,
доцент кафедри кібернетики та інформаційних технологій, ДБТУ,
м. Харків, Україна, vvdemina17@gmail.com, ORCID iD: 0000-0001-6467-5021

ДИНАМІЧНА ПРЕДМЕТНА ОБЛАСТЬ: МОДЕЛЮВАННЯ ТА ПОБУДОВА МЕТАПРОДУКЦІЙНИХ ВИВОДІВ

У статті проведено стислий аналіз наявних стандартних моделей представлення знань, а саме: логічних, продукційних, мережевих та фреймових. З цього переліку моделей представлення знань здійснено вибір деяких, які можна використовувати до формування єдиної, більш гнучкої та просунутої моделі. Розглянуто особливості функціонування цієї моделі за умов динамічної природи предметної області. Відзначено, що протиріччя баз знань – це особлива характеристика знань. Запропоновано загальний вигляд метапродукційних правил та умови їх застосування до вирішення завдань у динамічній предметній галузі. Запропоновано формат подання багаторівневої метапродукційної системи представлення знань. Запропоновано формат продукцій-моніторів, призначених до здійснення керування процесом виведення у багаторівневій метапродукційній системі представлення знань. Проведено попередній аналіз ситуацій, при яких необхідно використання метапродукційного підходу. Здійснено опис алгоритму виведення у багаторівневій метапродукційній системі представлення знань.

ПРОДУКЦІЯ, МЕТАПРОДУКЦІЯ, МЕТАПРОДУКЦІЙНІ ПРАВИЛА, СЕМАНТИЧНА ПРОДУКЦІЯ, ПОБУДОВА ВИВЕДЕНЬ, УЗАГАЛЬНЕНА СЕМАНТИЧНА ПРОДУКЦІЯ, МЕХАНІЗМ ВИВЕДЕННЯ

Марьин С. А., Побеженко И.О., Белова Т.Г., Дёміна В.М. Динамическая предметная область: моделирование и построение метапродукционных выводов. В статье проведен краткий анализ существующих, стандартных моделей представления знаний: логических, продукционных, сетевых, фреймовых. Из этого перечня осуществлен выбор некоторых из существующих моделей представлений знаний для формирования из них единой более гибкой и продвинутой модели представления знаний. Рассмотрены особенности функционирования этой модели в условиях динамической предметной области. Предложен общий вид метапродукционных правил и условия их применения к решению задач в динамической предметной области. Предложен формат представления многоуровневой метапродукционной системы представления знаний. Предложено формат продукций-мониторов, предназначенных для осуществления управления процессом вывода в многоуровневой метапродукционной системе представления знаний. Проведен предварительный анализ ситуаций, в которых необходимо использование метапродукционного подхода. Осуществлено описание алгоритма вывода в многоуровневой метапродукционной системе представления знаний.

ПРОДУКЦИЯ, МЕТАПРОДУКЦИЯ, МЕТАПРОДУКЦИОННЫЕ ПРАВИЛА, СЕМАНТИЧЕСКАЯ ПРОДУКЦИЯ, ПОСТРОЕНИЕ ВЫВОДОВ, ОБОБЩЕННАЯ СЕМАНТИЧЕСКАЯ ПРОДУКЦИЯ, МЕХАНИЗМ ВЫВОДА

Marin S., Pobizhenko I., Bilova T., Dyomina V. Dynamic subject area: modeling and construction of metaproductive conclusions. The article is devoted to brief analysis of existing and standard models of knowledge representation: logical, production, semantic network, frame. Some of the knowledge representation models are selected to form a more flexible and more advanced a knowledge representation. The article reviews the features of the functioning of a multilevel metaproduction system. This model of knowledge representation is functioned in a dynamic subject area. Metaproduction rules are proposed to solving problems in a dynamic area. Presentation format for a multi-level metaproduction is proposed. It is proposed to use production-monitors to control output process in a knowledge representation of multilevel metaproduction. It is proposed presentation format for special monitor-products. Analysis of situationns in which it is necessary to use the metaproduction approach is carried out. In this article were considered feature of constructing inference in a multilevel metaproduction model based on the use monitor-production. Definition of the inference algorithm in a multilevel metaproduction knowledge representation is created.

PRODUCTION, METAPRODUCTION, METAPRODUCTION RULES, SEMANTIC PRODUCTIONS, CONSTRUCTION CONCLUSION, GENERALIZED SEMANTIC PRODUCTS, MECHANISM OF CONCLUSION

Вступ

Інтелектуальні системи використовуються в різноманітних галузях сучасного суспільства, з метою вирішення різної складності та роду завдань. При їх реалізації застосовують методи створення більш складних і гнучких моделей, на базі простих, стандартних моделей представлення знань. Зробимо перелік груп подібних стандартних моделей.

Перша група — це продукційні моделі. Найпростіші, до того ж вони найчастіше використовувались при створенні експертних, а потім інтелектуальних систем. Друга група — логічні моделі. Їх базова особливість в безпосередньому використанні формул деякої логіки. Третя група — мережеві моделі. Однозначного визначення семантичної мережі донині не існує. Науковці в галузі штучного інтелекту під нею розуміють граф, що відображає сенс цілісного образу. Четверта — фреймові моделі. Фрейм – це мінімально можливий опис сутності будь-якої події, ситуації, процесу чи об'єкта. Поняття «мінімально можливе» означає, що при подальшому спрощенні опису втрачається його повнота, і воно перестає визначати ту одиницю знань, на яку було призначено. Уявлення знань на основі фреймів сприймається як один зі способів уявлення знань у ситуаціях.

В нашому випадку, для створення більш гнучкої моделі представлення знань було зроблено вибір із існуючих трьох моделей: логічної, мережевої та продукційної.

Апарат логіки було взято для використання у базовому описі стану семантичної мережі. Наприклад, опис мережі виглядає наступним чином:

$$S(t_0) = P_1 \wedge P_2 \wedge \ldots \wedge \overline{P_n} \ , \qquad (1)$$

де S – позначення стану семантичної мережі, t_0 – час у семантичній мережі, P_i – скорочена форма предикату, який описує наявність типу відносин між двома деякими об'єктами мережі. Предикат з запереченням описує відсутність певного типу відносин між двома деякими об'єктами мережі. Його повний вигляд наведено нижче:

$$P_i = P(A_i, L_k, A_j), \qquad (2)$$

де A_i, A_j – деякі об'єкти предметної галузі, а L_k – тип відносин між об'єктами предметної галузі.

Цей апарат також було використано для опису стану активного набору семантичних продукцій, та метапродукцій різних рівнів. Опис стану семантичних продукцій виглядає наступним чином:

$$S' = R_1 \wedge R_2 \wedge \ldots \wedge R_n ; \qquad (3)$$

Опис стану першого рівня метапродукційної моделі має вигляд аналогічний опису стану семантичної мережі:

$$S'' = R_1' \wedge R_2' \wedge \ldots \wedge R_n' ; \qquad (4)$$

Опис стану n-ого рівня метапродукційної моделі має наступний вигляд:

$$S^{n+1} = R_1^n \wedge R_2^n \wedge \ldots \wedge R_n^n ; \qquad (5)$$

Друга модель – семантичні мережі. Їх використання дає більше переваг перед іншими моделями представлення знань у моделюванні статичних галузей у виді повного переліку об'єктів, та відносин між ними. У ролі об'єктів можуть виступати реальні об'єкти предметної галузі, події, властивості, процеси. Відносини семантичної мережі мають або лінгвістичну, або логічну, або теоретико-множинну, або квантифіковану природу. Опис стану семантичної мережі – це перелік усіх її об'єктів та наявних відносин між ними. Цей стан легко описати використовуючи апарат логіки, кон'юнкцією предикатів, якими вказується або наявність, або відсутність відносин між об'єктами семантичної мережі.

Третя модель – продукційні системи. Треба відмітити, що вони достатньо прості як у створенні, так і у використанні в комп'ютерних системах. Їх переваги існують за рахунок наступного переліку особливостей. Перша – модульна організація знань. Друга – незалежність правил, які виражають цілком самостійні частини знань про предметну область. Третя – відокремлення предметних знань від керуючих, що дозволяє застосовувати різні стратегії управління та створювати загальні керуючі механізми для різних додатків. Останнє, однак головне, це простота внесення змін та модифікації предметної області.

Об'єднання цих трьох моделей дає низку очевидних переваг при моделюванні в умовах динамічних предметних областей. Воно дозволить усунути деякі явні недоліки кожної окремої моделі. Наприклад, статичність опису стану семантичної мережі – це явний недолік динамічної предметної області. Однак, застосування семантичних продукцій надасть змогу змінювати стан семантичної мережі достатньо легко. Наступний недолік продукційних правил – це присутність у поточному наборі конфліктуючих продукцій. Було запропоновано використання метапродукційних правил, яке дозволяє створювати безконфліктний набір продукційних правил.

Таким чином, у даній статті досліджуються особливості побудови виведення знань на базі використання продукційної системи з багаторівневими метапродукціями, які виконують функцію управління та безпосередньо впливають на перетворення структури відносин семантичної мережі. Після стислого дослідження предметної області було зроблено висновок про доцільність використання окремого типу продукцій, так званих продукцій-моніторів до управління процесом побудови виведення. Також досить стисло проведено аналіз ситуацій, які потребують метапродукційного застосування.

1. Постановка задачі

Загальною метою статті є дослідження проблеми побудови несуперечливого виведення в сучасних інтелектуальних системах на базі використання багаторівневої метапродукційної системи. Для рішення означеного завдання необхідно вирішення наступного переліку завдань:

– провести аналіз існуючих моделей представлення знань;

– створити відбір декількох моделей для формування з них більш гнучкої та потужної моделі представлення знань;

– переглянути особливості функціонування в умовах динамічної предметної області;

– переглянути загальний вигляд метапродукційних правил, умови та правила їх застосування до вирішення завдань у динамічному середовищі;

– запропонувати формат представлення багаторівневої метапродукційної системи;

– представити виведення на багаторівневій метпродукційній системі представлення знань, використання особливого виду продукцій-моніторів для управління виведенням;

– розглянути особливості застосування продукцій-моніторів до виведення в багаторівневій метапродукційній системі;

– описати алгоритм виведення в багаторівневій метапродукційній системі представлення знань.

2. Основний матеріал дослідження

Завдання метапродукційного управління полягає в підходах до моделювання динаміки змін знань про деяку предметну область. У нашому випадку такі зміни виявляються у зникненні з поточного набору одних правил та/або появі інших. Така поведінка набору правил вдало описується продукціями аналогічними семантичним продукціям, які «породжують» чі «вбивають». При цьому, якщо семантичні продукції задавали умови «народження» або «смерті» деяких відносин у мережі, то метапродукції повинні задавати умови «народження» або «смерті» самих семантичних продукцій.

Задамо загальний вигляд метапродукцій за аналогією з семантичними продукціями:

$$R'_i,: IF\left(S'_i,(t-\tau')\right)THEN\ \ R_k(t) \qquad (6)$$

$$R'_j,: IF\left(S'_j(t-\tau')\right)THEN\ \overline{R_k}(t) \qquad (7)$$

Підкреслимо, що τ' – тактовий час спрацьовування метапродукцій має бути більшим за час, протягом якого можуть відбутися зміни в динамічному стані самої семантичної мережі. Проте зазначимо, що фіксоване τ' під час вирішення завдань реальної складності вибрати неможливо. Це пов'язано з неперіодичністю або, у кращому випадку, з нестрогою періодичністю виникнення проблем виведення на продукціях нижнього рівня. Отже, запровадження жорсткого τ', що не змінюється, різко скоротить область застосування даної моделі. Крім того, збереження часу виведення на продукціях при використанні кількох або навіть одного рівня метапродукції є досить проблематичним, тому, надалі будемо розглядати τ' як деякий системний час, який дорівнює часу виконання одного циклу інтерпретатора. Відповідно $\tau' = n\tau$, де n – позитивне натуральне число. Роль тимчасового такту, що ініціалізує виведення, виконуватимуть продукції особливого типу, що модифікують сигнальну ділянку робочої пам'яті інтерпретатора. Докладніше цей механізм буде розглянуто нижче.

Оскільки τ' для всіх метапродукцій однаково, це дає можливість записати її у простішій формі:

$$\left(R'_i,\ F'_i\right): IF\left(S'_i\right)\ THEN\ R_k \qquad (8)$$

$$\left(R'_j,\ F'_j\right): IF\left(S'_j\right)\ THEN\overline{R_k} \qquad (9)$$

дде до двокрапки в круглих дужках знаходиться передумова метапродукції, що містить логічну змінну R', котра вказує на активність метапродукції та логічне вираження F', що описує клас причини ініціалізації метапродукційного рівня; S' – умова застосування ядра метапродукції, або диз'юнкція станів набору продукцій, що визначають «народження» («смерть») на наступному такті продукції R_k. Перша метапродукція є умовою «народження» (появи, активізації) у наборі продукцій продукції R_k. Друга метапродукція – умова «смерті» (зникнення) у наборі продукцій продукції R_k.

Наведені метаправила в простій формі задають причинно-наслідковий зв'язок між двома послідовними станами набору правил. Слід зазначити, що з першої метапродукції $S'_i = \bar{R}_k \wedge S'_i$. Це означає, що має сенс розглядати умову «народження» лише для відсутньої продукції. Аналогічно: $S'_j = R_k \wedge S'_j$. Останній запис означає, що умову «смерті» можна розглядати лише для «живої» продукції. Розглянемо наприклад метаправило:

$$\left(R'_1, F'_1\right): IF\left(\left(R_2 \vee R_3 \wedge \bar{R}_4\right) \wedge R_1\right)\ THEN\ \ \bar{R}_1 \qquad (10)$$

Ця метапродукція спрацює, якщо вона жива ($R'_1 = 1$) і призначена для вирішення проблем класу F'_1. Її можна проінтерпретувати таким чином: правило F'_1 зникне з набору правил, якщо на попередньому такті в цьому наборі було правило R_2 або було правило R_3 та відсутнє правило R_4.

При моделюванні більш складних об'єктів, може виникнути ситуація, коли множина метапродукцій буде завеликою та буде потрібно шукати засоби управління метапродукціями. У цьому випадку, має сенс, запровадити поняття метапродукції другого порядку, яка буде здатна «породжувати» або «вбивати» метапродукцію першого порядку. Наприклад:

$$\left(R'_1, F''_1\right): IF\left(R'_1 \wedge R'_2\right)THEN\ \overline{R'_1} \qquad (11)$$

де в передумові: R'_1 – мітка метаправила другого рівня, а F''_1 – стан процесу виведення метаправил першого рівня.

Аналогічно можна розглядати метапродукції n-го рівня, як правила, що керують метапродукціями n-1-го рівня:

$$\left(R^n_i,\ F^n_i\right):\ IF\left(S^n_i\right)\ THEN\ R^{n-1}_k \qquad (12)$$

$$\left(R^n_j,\ F^n_j\right):\ IF\left(S^n_j\right)\ \ THEN\overline{R^{n-1}_k} \qquad (13)$$

де n – порядок метапродукції. Рівні метапродукцій, подібно до рівнів деталізації, надають спосіб абстра-

гування складності законів динамічної предметної області. Отже, коли зростає складність поведінки модельованого мережею об'єкта, зростає кількість рівнів метаправил, які необхідні до його адекватного опису. Метапродукції найвищого рівня – статичні. Тобто набір активних продукцій цього рівня зовсім не змінюється у процесі виведення. Крім цього, багатошарова структура продукційної системи висуває додаткові вимоги до механізмів виведення.

Необхідний формальний засіб, який би дозволив регламентувати відносини між шарами моделі. Крім того, оскільки обсяг кожного продукційного рівня може досягати декількох сотень правил, у цьому випадку, необхідно використання гнучкого апарату управління, який легко модифікується та має можливість досить легко керувати процесом вибору рівня виведення. У цій статті запропоновано в якості подібного засобу використовувати продукції-монітори. Принцип визначення необхідності виведення на метапродукціях є досить простим. Якщо при інтерпретації набору продукцій певного рівня виникли проблеми, які не вирішуються в рамках цього рівня, тоді ініціювати роботу продукцій/метапродукцій рівнем вище. Оцінка проблеми формується з урахуванням двох аспектів: синтаксичного та семантичного. Інформація, що надходить від інтерпретатора, є джерелом для отримання висновку про синтаксичну складову проблеми.

Стисло проаналізуємо ситуації, які можна розцінювати як такі, що не відповідають нормальному функціонуванню продукційної системи:

1) у наборі активних продукцій/метапродукцій є конфліктуючі продукції/метапродукції;

2) зміни семантичної мережі мають циклічний характер;

3) вичерпано максимально допустиму глибину виведення;

4) у наборі активних продукцій/метапродукцій немає жодної застосованої до поточного стану бази фактів («глухий кут»).

Перша із зазначених причин найбільш повно досліджена фахівцями з штучного інтелекту. А застосування метапродукційної стратегії для створення безконфліктного набору знань можна вважати традиційним. Семантичні метапродукції дозволяють уникати протиріч під час виведення шляхом відключення частини продукцій. За допомогою метапродукцій формується безконфліктний набір активних продукційних правил, тобто фактично вибирається одна продукція з кількох можливих.

Циклічна зміна елементів робочої пам'яті (у разі – це семантична мережа), під час вирішення певного класу завдань, (наприклад, завдань орієнтованих на пошук ланцюжка перетворень, яка веде до деякого цільового стану мережі) не допустима. Розпізнавання таких проблем зазвичай покладається на інтерпретатор. Набір його функцій поповнюється для забезпечення засобів доступу до інформації про попередні кроки виведення (так звана «історія виведення»).

Наступна причина зупинки системи пов'язана з існуванням деякого кінцевого числа кроків виведення. Якщо минуло кілька тактів і досягнуто цільовий стан, то, у цьому разі, говорять про нездатність системи визначити рішення. Використання метапродукційних шарів, що змінюють поточний набір правил системи, дає можливість ще раз спробувати знайти рішення на основі нового модифікованого набору семантичних продукцій.

Остання синтаксично нерозв'язна проблема пов'язана з виникненням такого стану семантичної мережі, коли жодна активна продукція неспроможна спрацювати, тобто досягнуто один з термінальних станів. Якщо при цьому системою не знайдено рішення, таку ситуацію можна охарактеризувати як невдачу системи. У цьому, як і в попередньому випадку, перебудова поточного активного набору продукцій дає шанс продовжити пошук рішення.

Завдання правил-моніторів, при ідентифікації будь-якої із зазначених проблем інтерпретатором, полягає у формуванні сигнального елемента робочої пам'яті та, відповідно, ініціалізації процесу виведення на метапродукційному рівні. Більш того, у деяких випадках, продукції-монітори можуть розширювати можливості інтерпретатора щодо ідентифікації проблем. Наприклад, під час відстеження циклів, монітори дозволяють визначати циклічність виникнення як конкретного стану мережі, так і циклічність появи деякого класу станів.

Зазначені вище ситуації, що виникають у процесі виведення, можна розглядати з погляду траєкторії зміни станів семантичної мережі (тобто форми життєвого циклу об'єкта, що моделюється). Цикл і лінійна послідовність формалізують елементарні типи динамічної поведінки об'єктів, що моделюються. Конфлікти дозволяють комбінувати ці елементарні типи шляхом звернення до метапродукційних рівнів. Таким чином, метапродукції можна розглядати як засіб для моделювання динаміки продукцій даній предметній області. Більша кількість метапродукційних шарів дозволяє моделювати складніші форми траєкторії станів системи.

Наведений вище аналіз причин, що вимагають метапродукційного виведення, торкнувся лише їх синтаксичного аспекту. Тепер розглянемо ситуації, коли семантика предметної області потребує зміни поточного набору/метапродукцій. Слід зазначити, що виникнення таких ситуацій найхарактерніше для тих завдань, в яких продукції розглядаються як оператори, що змінюють стан предметної області. Причини виникнення подібних ситуацій криються

у складній природі причинних зв'язків та неможливості їхнього повного обліку на етапі формування стратегії досягнення мети.

Роль продукцій-моніторів у процесі прийняття рішення полягає у визначенні ситуацій предметної області, у разі виникнення яких необхідно змінити поточний тип поведінки чи стратегію. Продемонструємо її на прикладі ухвалення рішення спортсменом-марафонцем. Отже, маємо інтелектуальну систему, перед якою стоїть завдання якнайшвидше досягти фінішної межі. Процес вибору стратегії поведінки, у цьому випадку, гранично простий: бігти найкоротшим шляхом у напрямку мети (тобто активізувати ті продукційні правила, які формалізують переміщення об'єкта до наміченої межі). Однак, слідування обраній лінії поведінки може виявитися неможливим через поступове проявлення тих причинно-наслідкових зв'язків, які не були враховані при початковому виборі способу досягнення мети, але які можуть перешкоджати її досягненню. Прикладами подібних зв'язків можуть бути: «біг – швидка витрата енергії – голод – неможливість продовжувати марафон», «біг – посилене потовиділення – зневоднення – неможливість продовжувати марафон», «біг – посилена робота м'язів – підвищення температури тіла – неможливість продовжувати марафон» тощо. Як діє у цьому випадку спортсмен-марафонець? Він коригує свою поведінку в процесі вирішення основного завдання. Якщо раніше марафонець суворо слідував стратегії «бігти найкоротшим шляхом» і байдуже пробігав повз столиків з соками і продуктами, розставленими по ходу траси, то тепер він пригальмовує біля найближчого столика, щоб попити або поїсти (тобто продукційна модель, що описує його поведінку, повинна поповнитися правилами, які стосуються процесів прийняття їжі). Після того, як все, що заважає швидкому досягненню мети подолано, марафонець повертається до попередньої моделі поведінки.

Таким чином, продукції-монітори реалізують постійний контроль над процесом вирішення завдань, відстежуючи ситуації, які перешкоджають досягненню мети, ініціалізуючи виведення на метапродукціях з метою усунення цих перешкод.

Розглянемо формальне представлення продукцій-моніторів. Їхня умовна частина повинна включати:

1) опис множини станів семантичної мережі, які вимагають зміни поточного набору продукцій;

2) поточну оцінку процесу виведення. Заключна частина продукції призначена для модифікації поточної оцінки процесу виведення.

Загальний вид продукцій-моніторів:

$$IF\ (S_i \wedge F_i')\ \ THEN\ C_k' \qquad (14)$$

$$IF(S_j \wedge F_j')\ THEN\ \bar{C}_m' \qquad (15)$$

де умова застосування продукції представлена кон'юнкцією деякого стану семантичної мережі (S_i, S_j) та поточного стану процесу виведення $(F_i,\ F_j)$. Перша продукція розпізнає ситуації, за яких проблема C_k' виникає, а друга описує закон усунення проблеми $\bar{C}_m'$.

Зазначимо, відсутність логічної змінної, яка вказує на активність правила. Це пов'язано з принциповою неможливістю виникнення конфліктних ситуацій між продукціями-моніторами (йдеться тільки про синтаксичні конфлікти, семантичні цілком можливо матимуть місце). З усієї сукупності продукцій-моніторів взаємозалежні виведення мають пари продукцій, що включають установку або зняття маркера, який ідентифікує певний клас проблем. Проте, умовні частини таких продукцій не можуть перетинатися, оскільки встановлення маркера передбачає його відсутність на попередньому кроці виведення, а відключення, відповідно, його наявність у попередній оцінці. Отже, можливість появи конфліктів між продукціями-моніторами повністю виключається.

Необхідність у вирішенні конфліктів різного роду на етапі виведення притаманна всім рівням метапродукційної моделі. Отже, множина продукцій-моніторів, як і семантичні метапродукції, має багаторівневу структуру. Проте розбиття її на рівні відрізняється від метапродукційного. Як зазначалося вище, ознакою, за якою визначається рівень метапродукції, є природа об'єктів в умовній і заключній частинах. Якщо орієнтуватися на такі засади розподілу, то виявиться, що продукції-монітори належать відразу двом рівням (або можна говорити, що вони розташовані між двома рівнями).

Продукції-монітори, що ініціалізують висновок на метапродукції n-ного шару, можна представити наступним чином:

$$IF\left(S_i^{(n-1)} \wedge F_i^{(n)}\right) THEN C_k^{(n)} \qquad (16)$$

$$IF\left(S_j^{(n-1)} \wedge F_j^{(n)}\right) THEN \overline{C_k^{(n)}} \qquad (17)$$

Багато продукцій-моніторів певного рівня будемо позначати MR^l, де l – номер рівня продукцій-моніторів.

Завдання керування виведенням у багаторівневих моделях представлення знань складається з рішення двох завдань. Перше з яких пов'язано з проблемами "внутрішньорівневого" виведення, а друге охоплює питання перемикання між рівнями продукцій/метапродукцій.

Залучення вищих шарів метаправил у процес виведення ґрунтується на виникненні синтаксичних та семантичних конфліктів на попередніх рівнях. Синтаксичні конфлікти визначаються інтерпретатором та означають важливу неможливість продовження процесу виведення при поточному наборі продукції. Семантичні конфлікти визначаються набором продукцій-моніторів і вказують, що зміни, що відбулися з предметною областю, вимагають модифікації активного набору продукцій. Формально при

семантичному конфлікті жодних перешкод для продовження процесу виведення немає. Відсутність конфліктів дозволяє вести виведення без змін поточного набору продукцій.

Таким чином, проблема визначення рівня виведення вирішується шляхом оцінки процесу виведення та його результатів на кожному продукційному рівні, на кожному кроці виведення. Перший метапродукційний рівень, виведення на якому відбулося без конфліктів називатимемо рівнем виведення. Представимо описану процедуру виведення у більш строгій алгоритмічній формі:

{виведення починається з базового рівня правил }
level:=1;
While (спрацювавших правил останова нема) do
  begin
    зіставити умовні частини активних правил $S_i^{(level)}$ з фактами $S_j^{(level-1)}$;
if (виявлено синтаксичний конфлікт) then
    модифікувати $F_i^{(level)}$;
Виведення на безлічі MR^{Level} продукций-мониторов;
if (виявлено семантичний конфлікт) then модифікувати $F_i^{(level)}$;
if ($F_i^{(level)} \neq F_{i-1}^{(level)}$ then
  level:=level+1;
else
  if (Level≠1) then
            level:=level-1;
    end;

Зупинення процесу виведення відбувається або після досягнення цільового стану основи фактів (стану семантичної мережі), або після перевищення числа кроків виведення.

Висновки

Таким чином, у статті було розглянуто наступне коло питань:

1. Стисло проведено огляд наявних стандартних моделей представлення знань.

2. Створено відбір декількох моделей для формування з них більш гнучкої, потужної та складної моделі представлення знань.

3. Розглянуто особливості функціонування цієї моделі в умовах динамічної предметної області.

4. Розглянуто загальний вигляд метапродукційних правил, умови та правила їх застосування до вирішення завдань у динамічному середовищі.

5. Стисло переглянуто особливості функціонування багаторівневої метапродукційної системи, запропоновано її формат представлення.

6. Запропоновано використання особливого виду продукцій-моніторів для управління виведенням на багаторівневій метапродукційній системі представлення знань.

7. Запропоновано особливий формат продукцій-моніторів.

8. Стисло розглянуто особливості застосування моніторів до виведення в багаторівневій метапродукційній системі знань.

9. Побудовано та створено формальний опис алгоритму виведення в багаторівневій метапродукційній системі представлення знань.

Список літератури:

[1] *Дударь З. В.,* Калиниченко О. В., Шабанов-Кушнаренко С. Ю. О методе и задачах теории интеллекта. I. // Радиоэлектроника и информатика. 2000. № 2. С. 112 – 122.

[2] *Котов И. А.,* Суворов А. И., Сердюк А. Ю. Разработка методов унификации структурно-логической модели метазнаний для управления эволюцией онтологий интеллектуальных систем URL: https://core.ac.uk/display/288839718.pdf

[3] *Шабанов-Кушнаренко С. Ю.,* Кудхаир Абед Тамер, Лещинская И. А. Предикатный подход к формализации неявных знаний // Системи обробки інформації Н. 2013. Вип. 9. С. 113–116.

[4] *Ligeza A.* Knowledge Representation and Inference for Analysis and Design of Database and Tabular Rule-Based Systems // Computer Science. 2001. Vol. 3, Issue 1. P. 13–60.

[5] *Morkun V.,* Tcvirkun S. Investigation of methods of fuzzy clustering for determining ore types // Metallurgical and Mining Industry. 2014. Issue 5. P. 12–15. URL: http://www.metaljournal.com.ua/assets/Journal/3-MorkunTs.pdf

[6] *Nasrollahi S. N.,* Mokhtari H., Seyedein M. Meta-analysis: An Approach to Synthesizing and Evaluating Research on Knowledge and Information Science // Iranian Journal of Information Processing & Management. 2011. Vol. 29, Issue 2. P. 293 – 316.

[7] Open Data as a key factor for developing expert systems: a perspective from Spain / Rodriguez-Rojas L. A., Cueva-Lovelle J. M., Tarazona-Bermudez G. M., Montenegro-Marin C. E. // International Journal of Interactive Multimedia and To Artificial Intelligence. 2013. Vol. 2, Issue 2. P. 51. doi: https://doi.org/10.9781/ ijimai.2013.226

Надійшла до редколегії 09.11.2021

УДК 65.011.56 DOi 10.30837/ bi.2021.2(97).08

Н.М. Боргест[1], А.И. Повзун[2], Г.Г. Четвериков[3]

[1]Кандидат технических наук, профессор,
Самарский университет, Россия, borgest@yandex.ru

[2]Кандидат технических наук, доцент, Донецький национальный технический университет,
Украина, povzun.aleksey@gmail.com

[3]Доктор технических наук, профессор, Харьковский национальный университет радиоэлектроники,
Украина, ORCID iD: 0000-0001-5293-5842, chetvergg@gmail.com

МАТЕМАТИЧЕСКИЕ АСПЕКТЫ БАЗОВОЙ ОНТОЛОГИИ В ЗАДАЧАХ ПЛАНИРОВАНИЯ ПРОИЗВОДСТВА МАШИНОСТРОИТЕЛЬНЫХ ПРЕДПРИЯТИЙ

На основе онтологического анализа рассмотрены задачи управления машиностроительным предприятием и, в частности, задачи планирования производства. Проанализированы подходы и принципы создания онтологий машиностроительного предприятия. Особое внимание уделяется мультиагентным технологиям и интерсубъективному подходу при описании онтологических свойств акторов на предприятии. В статье рассмотрены задачи онтологии применительно к управлению машиностроительным предприятием и, в частности, к планированию производства. Проанализированы подходы к созданию онтологии предприятия. Разработанный фрагмент базовой онтологии предприятия позволяет наглядно представить суть взаимодействий базовых сущностей, обосновано подойти к формированию адекватных критериев и условий их взаимодействия.

ОНТОЛОГИЯ, ОНТОЛОГИЯ ВЕРХНЕГО УРОВНЯ, АГЕНТ, МУЛЬТИАГЕНТНЫЙ ПОДХОД, ИНТЕРСУБЪЕКТНАЯ ТЕОРИЯ

Боргест Н.М., Повзун А.И., Четвериков Г.Г. Математичні аспекти базової онтології в задачах планування виробництва машинобудівельних підприємств. На основі онтологічного аналізу розглянуто завдання управління машинобудівним підприємством і, зокрема, завдання планування виробництва. Проаналізовано підходи і принципи створення онтологій машинобудівного підприємства. Особлива увага приділяється мультиагентний технологіям і інтерсуб'єктивності підходу при описі онтологічних властивостей акторів на підприємстві. У статті розглянуті завдання онтології стосовно до управління машинобудівним підприємством і, зокрема, плануванням виробництва. Проаналізовано підходи до створення онтологій. Представлений фрагмент базової онтології підприємства дозволяє наочно уявити суть взаємодій базових сутностей, обгрунтовано підійти до формування адекватних критеріїв і умови їх взаємодії.

ОНТОЛОГІЯ, ОНТОЛОГІЯ ВЕРХНЬОГО РІВНЯ, АГЕНТ, МУЛЬТИАГЕНТНИЙ ПІДХІД, ІНТЕРСУБ'ЄКТНА ТЕОРІЯ

Borgest N.M., Povzun A.I., Chetverykov G.G. Mathematical aspects of the basic ontology in the problems of production planning for machine-building enterprises On the basis of ontological analysis, the problems of management of a machine-building enterprise and, in particular, the problems of production planning are considered. The approaches and principles of creating ontologies for a machine-building enterprise are analyzed. Special attention is paid to multiagent technologies and the intersubjective approach when describing the ontological properties of actors in an enterprise. The article describes the problems of ontology in relation to the management of an engineering facility and production planning. The approaches to ontology creation are analyzed. Given representation of the basic ontology of the enterprise allows to visualize the nature of the interactions of basic entities, grounded approach to the formation of adequate criteria and conditions of their interaction.

ONTOLOGY, TOP-LEVEL ONTOLOGY, AGENT, MULTI-AGENT APPROACH, INTERSUBJECT THEORY

Введение

Развитие информационных и коммуникационных технологий за последние десятилетия определило применение междисциплинарных подходов к построению и организации работы информационных систем. В связи с этим возникла проблема обеспечения когнитивной прозрачности при сохранении точности формальной семантики в отображении все более сложного и гетерогенного информационного поля. Решением этой проблемы может быть онтология, общая теория типов объектов и отношений, составляющих конкретные предметные области, иначе концептуальная модель, позволяющая исследователям сфокусироваться на смысле информации, без привязки к конкретным форматам и языкам представления данных.

1. Постановка проблемы

Усложнение IT инфраструктуры и архитектуры баз данных предприятия диктует необходимость поиска новых подходов к организации хранения информации, перехода к семантически интегрированным базам знаний. Онтологический подход к организации данных предлагает методы решения данной проблемы, однако практическое применение общих онтологий сопряжено с рядом проблем внедрения. Решением части этих проблем может стать реализация специализированной базовой онтологии

предприятия, изначально предназначенной для решения практических задач современного машиностроительного предприятия.

2. Понятия и принципы

В философии под онтологией понимается наука о сущем, о видах и структуре объектов, свойств, событий, процессов и отношений в различных предметных областях. Исследуя онтологию конкретного предприятия, можно говорить о задаче интеграции знаний на этом предприятии, как о высшей форме представления знания, и абсолютно недостижимой в силу ограниченности возможностей человека [1]. С учетом имеющихся программно-аппаратных ограничений, под онтологией в практическом смысле подразумевают формализацию и концептуализацию знаний в конкретной предметной области. Формализация знаний предполагает их классификацию. На сегодняшний день не существует общего мнения о том, какие базовые концепты должны составлять онтологию, однако большинство исследователей сходятся во мнении, что онтология должна состоять из классов сущностей предметной области, свойств этих классов, связей между этими классами и утверждений, построенных из этих классов, их свойств и связей между ними.

Онтологические подходы становятся все более популярными, расширяется сфера их применения – современные онтологические модели позволяют описывать нечеткие предметные области [2].

Существует два основных подхода к проектированию онтологии: восходящее и нисходящее проектирование [3]. Сущность восходящего проектирования заключается в последовательном описании предметной области, начиная с самых частных концептов, например, для машиностроительного предприятия это может быть описанием рабочего места токаря, с последующим объединением получающихся «мини-онтологий» в общую систему. Онтологии, получающиеся таким путем, являются узкоспециализированными и трудно применимыми в смежных предметных областях [4]. Нисходящее проектирование онтологий заключается в предварительном построении онтологии на высоком уровне абстракции, где бы описывались наиболее общие, базовые концепты, такие как «класс», «свойство класса», «отношение», общие для многих предметных областей, с последующим доопределением концептов по уже имеющейся классификации. Такие онтологии высокого уровня абстракции называют онтологиями верхнего уровня или «top level ontology» в зарубежной литературе [5].

Онтологии верхнего уровня представляют систему, в рамках которой различные системы могут использовать общую базу знаний, позволяя при этом объединять множество специализированных онтологий более низкого уровня [6]. Концепты, определяемые такой онтологией, являются базовыми и универсальными для множества предметных областей, что позволяет обеспечивать целостность и непротиворечивость совокупной информационной системы [7]. Стандартные онтологии верхнего уровня иногда называют универсальными онтологиями [8].

На сегодняшний день создано множество онтологических систем верхнего уровня. Среди наиболее известных систем можно назвать SUMO, онтологию Sowa [9], Dolce [10], Clip [11] и ISO 15926-2 [12]. Вышеперечисленные онтологии объединяет возможность определения ключевых концептов через описание их поведения и сценариев использования. Затем эти концепты определяются на основании более общих концептов, заложенных в онтологии верхнего уровня [6]. Такой подход позволяет с одной стороны, избежать необходимости каждый раз «изобретать велосипед» при определении классов сущностей, а с другой предоставляет широкие возможности адаптации онтологии под конкретную задачу и в значительной мере упрощает её поддержку.

На рис. 1 представлена многоуровневая структура онтологии машиностроительного предприятия. Онтологии верхнего и нижнего уровней создаются экспертами в предметной области в тесном сотрудничестве со специалистами-онтологами, тогда как задача наполнения онтологии концептами нижнего уровня решается пользователями без привлечения дополнительных специалистов. На практике, наиболее хорошо зарекомендовавшим себя методом является наполнение онтологии через специально спроектированные формы-шаблоны, позволяющие человеку без навыков программирования или представлений о том, как устроена онтологическая база знаний, эффективно решать задачу её наполнения [11].

Нисходящее проектирование онтологий имеет ряд преимуществ, например, наличие общей онтологии верхнего уровня, позволяет легко обеспечивать интероперабельность между отдельными «дочерними» онтологиями в рамках общей системы понятий.

Для обеспечения и поддержания семантической интеграции информационных ресурсов предприятия используемая онтология должна соответствовать следующим основным принципам [13, 14]:

1. Принцип прозрачности: онтология, предназначенная для общего использования должна быть понятна для персонала, знакомого с принципами построения и работы онтологий, что обеспечивает поддерживаемость и развиваемость онтологической системы.

2. Принцип открытости: использование открытых стандартов значительно упрощает задачу системной интеграции программных комплексов предприятия.

Рис. 1. Структура онтологии машиностроительного предприятия

3. Принцип повторного использования доступных ресурсов: в случае наличия готовых онтологий для конкретных предметных областей (например, онтология склада готовой продукции, онтология инструментального цеха), эффективнее использовать проверенные готовые решения, чем создавать собственный продукт с дублирующим функционалом.

4. Принцип консистентности: онтология предприятия должна обеспечивать целостность, органическую взаимосвязанность и согласованность всех элементов информационной системы предприятия.

5. Принцип поддержки версионности продукции: некоторые виды наукоемкой машиностроительной продукции имеют различный элементный состав от изделия к изделию, что должно находить свое отражение в онтологии.

6. Принцип разумной достаточности: если определение не содержит полезной информации по применимости концепта, оно не должно использоваться.

С учетом вышесказанного можно сделать следующий вывод – онтология, претендующая на роль базовой онтологии машиностроения должна, с одной стороны, быть достаточно абстрактной для охвата всей потребной предметной области, с другой – быть в достаточной степени специализированной для машиностроения с целью снижения затрат и рисков при внедрении.

3. Семантическая сеть

Современные системы управления – это программные комплексы высочайшей сложности. Постоянный рост количества элементов таких систем заставил специалистов искать принципиально новые подходы к автоматизированному управлению. Одним из наиболее перспективных стал зародившийся в 1980-х годах метод, получивший название многоагентный или мультиагентный подход. Сущность данного подхода заключается в переходе от централизованного управления «сверху» к самоорганизующейся системе, образованной несколькими взаимодействующими интеллектуальными агентами, как программными, так и живыми, для которых описаны их цели и правила взаимодействия. Под интеллектуальностью в этом случае понимается способность агентов обучаться и приспосабливаться.

Задачей онтологии в этом случае является обеспечение взаимодействия агентов, не обязательно оперирующих в рамках одной общей теории. Агент действует в рамках онтологии, если его наблюдаемые действия укладываются в классификацию этой онтологии. Таким образом, онтология определяет «словарь» общения между агентами. Онтологическое согласование может происходить на разных уровнях абстракции, то есть агенты, оперирующие в близких предметных областях, обмениваются более конкретными данными, тогда как слабосвязанные агенты при обмене оперируют сущностями типа «класс», «отношение» и т.п. Использование онтологий позволяет достичь согласованности между агентами, однако не гарантирует полноты описаний. Множественность онтологий реального мира в данном контексте обусловлена различиями внутренних интерпретаций предметных областей разными исследователями [15–16].

На сегодняшний день одной из наиболее перспективных форм концептуализации знаний признаны семантические сети, представляющие собой модель

предметной области в виде ориентированного графа, вершины которого соответствуют объектам предметной области, а дуги задают отношения между ними.

На рис. 2 в качестве примера описания информационной среды предметной области машиностроительного предприятия представлена общая схема взаимодействия между агентом заказа и агентами рабочих мест в системе внутрицехового планирования завода. Агент заказа содержит полную информацию о заказе, включая технологию, трудоемкость, стоимость и пр. Агент, выполняющий функцию рабочего, представляет собой онтологические свойства исполнителя, такие как разряд, специальность, загруженность и пр. Итогом взаимодействия агентов, обычно интерпретируемое в мультиагентных технологиях как матчинг, является составленное расписание выполнения заказов.

Онтологические модели в силу своей природы, как правило, достаточно объемны. «Масштабный фактор» – возрастание сложности системы из-за увеличения количества связей между её частями – диктует необходимость в новых подходах к управлению сложными системами. Мультиагентный подход, позволяет значительно сократить количество связей в системе, являясь одним из наиболее перспективных способов управления сложными распределенными системами.

Мультиагентный подход при создании интеллектуальных систем основывается на построении комплекса, состоящего из множества агентов. Таким образом, все управление системой осуществляется коллективом агентов, которые адаптируются под решение конкретной задачи.

Мультиагентные системы значительно отличаются от классических систем. В настоящее время основной акцент в разработке таких систем ставится на раздельной обработке и «социальном» поведении агентов. Интересной особенностью мультиагентной системы является возможность представления человека в качестве агента, что отрывает широкие возможности при постепенном переходе от операторов к компьютерным агентам.

Стоит отметить, что «агент» – более общее понятие, чем система или ее элементы. Это позволяет при необходимости представлять любую часть сложной системы в качестве агента. Представленная модель

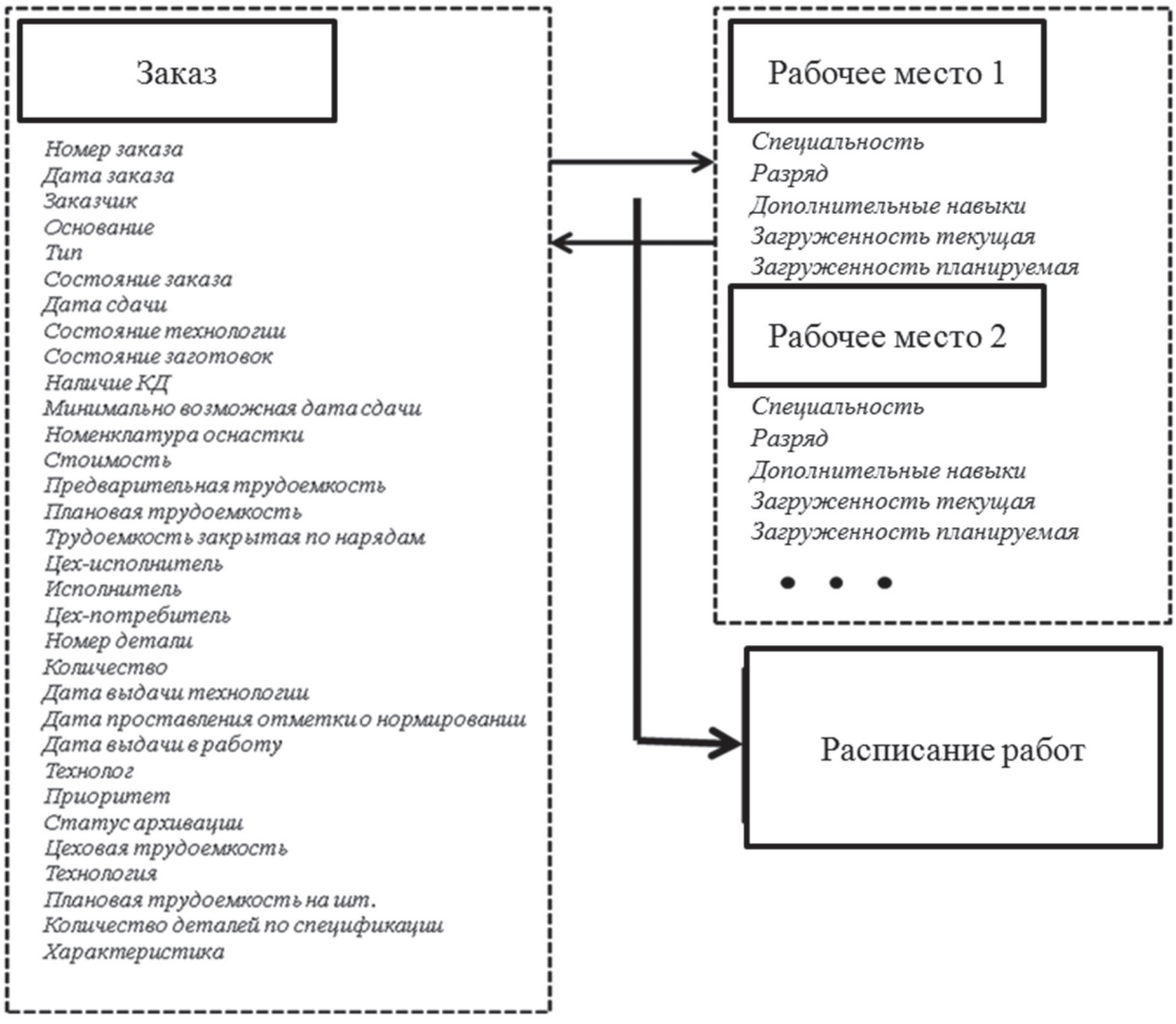


Рис. 2. Схема взаимодействия агентов

пригодна для машинной обработки, что значительно расширяет возможности её использования.

Агенты взаимодействуют между собой в рамках общей внутрицеховой онтологии, которая в свою очередь входит в более общую онтологию предприятия. Результатом работы системы является расписание работ внутри цеха, фрагмент которого приведен в табл. 1. Каждая деталь или сборочная единица (ДСЕ) имеет собственный набор необходимых операций, для которых важна последовательность. Для каждой операции необходимо найти исполнителя и обеспечить правильную технологическую последовательность операций при выполнении ряда условий, например, возможно равномерного распределения нагрузки для рабочих цеха.

Схема взаимодействия агентов приведена на рис. 2. Здесь заказ – более общее понятие, чем ДСЕ, так как один заказ может включать в себя несколько ДСЕ. Визуализация результата может быть выполнена в форме диаграммы Ганта, как показано на рис. 3.

Знания об объекте соотносится не только с особенностями его взаимодействия со средствами наблюдения, но и с ценностно-целевыми структурами деятельности субъекта. Таким образом, можно вести речь о необходимости создания интерсубъективной теории [14], представляющей собой интеграционную платформу для достижения консенсуса неоднородными агентами относительно способа регулирования проблемной ситуации. Включение человека, его онтологических свойств: восприятия, способностей, возможностей, мотивации и потребностей, в систему, в процесс управления предприятием, существенно расширяет потенциал возможностей самой системы.

Интерсубъективный подход в сочетании с мультиагентным формируют не только иные парадигмы взаимодействия сущностей в управлении предприятием, но соответственно иные критерии такого взаимодействия и сами модели, описывающие бизнес процессы.

Заключение

Переход к управлению на основе мультиагентных технологий, базирующийся на онтологическом подходе при описании предметной области, позволяет современному предприятию переходить к экономике реального времени, повышает эффективность использования ресурсов, снижает затраты времени. Реализация мультиагентных технологий поддерживает непрерывное планирование в реальном времени с немедленной реакцией на события, позволяет создать масштабируемую платформу для решения задач высокой сложности. Реализация интероперабельности агентов диктует необходимость в обеспечении

Таблица 1

Фрагмент расписания работ внутри цеха

Название	Номер	Исполнитель	Время начала	ДСЕ
Слесарная	1	Богатов Сергей Викторович	24.01.2013 6:25	74774/113[Приспособление]
Фрезерная	6	Лукшина Надежда Альбертовна	24.01.2013 9:50	74774/113[Приспособление]
Токарная	11	Беребердин Михаил Иванович	24.01.2013 12:20	74774/113[Приспособление]
Слесарная	15	Ерофеев Геннадий Викторович	24.01.2013 15:50	74774/113[Приспособление]
Контроль	15	Сергеев Виктор Борисович	24.01.2013 18:10	74774/113[Приспособление]

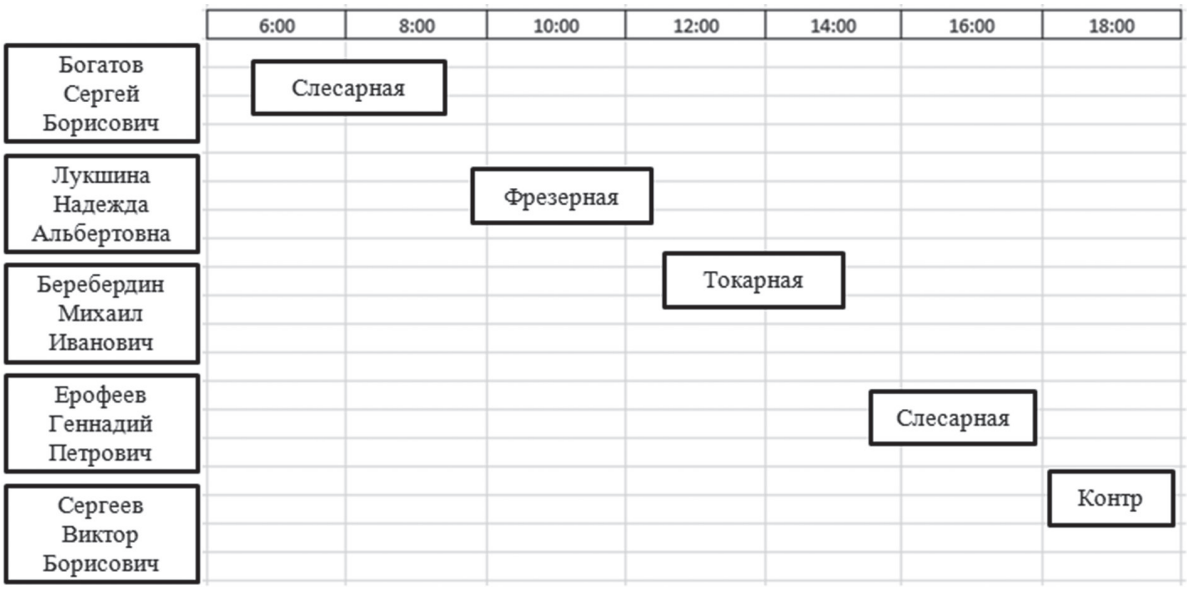


Рис. 3. Фрагмент цехового расписания в форме диаграммы Ганта

связанности онтологий их деятельности, что может быть достигнуто в рамках специализированной онтологии верхнего уровня для машиностроительной отрасли. Создание подобных сущностных прикладных моделей позволяет повысить прозрачность принятия решения и ответственность за их исполнения, а также в значительной мере упростить сами задачи ситуационного управления и планирования на предприятии.

Список литературы

[1] *Mead, G. H.* The individual and the social self: Unpublished work of George Herbert Mead (D. L. Miller, Ed.). Chicago: University of Chicago Press, 1982.

[2] *Потапова Е.В.* Модель лингвистической онтологии предметной области с нечеткими семантическими состояниями терминов /Е.В. Потапова// Бионика интеллекта 2012 №2 (79). С. 95-102

[3] Building the Semantic Web for the Process Industries using RDF, OWL, and SPARQL for the Integration, Sharing, Exchange, and Hand-over of Plant Lifecycle Information on the basis of ISO 15926.: An all-in explanation of ISO 15926, from data model to implementation.

[4] *Yang, A.* and W. Marquardt, 2004, An Ontology-based Approach to Conceptual Process Modelling. Proceedings of ESCAPE-14, Portugal.

[5] *N. Casellas,* M. Blázquez, OPJK into PROTON: Legal domain ontology integration into an upperlevel ontology. Proceedings of the 3rd International Workshop on Regulatory Ontologies (WORM 2005), volume 3762 ofLecture Notes in Computer Science, pages 846–855. Springer, 2005.

[6] *G. Nagypál* and J. Lemcke. A business data ontology. Data, Information and Process Integration with Semantic Web Services Project, FP6 U 507483, Deliverable D3.3, 2005.

[7] *A. Abdulaev*. Reality, Universal Ontology and Knowledge Systems: Toward the Intelligent World, 2008. 306 p.

[8] Jan SCHEFFCZYK, Adam PEASE, Michael ELLSWORTH. Linking FrameNet to the Suggested Upper Merged Ontology, 2008.

[9] *Sowa, J.*, 2000, Knowledge Representation: logical, philosophical, and computational foundations. Brooks/Cole

[10] *Gangemi A.*, N. Guarino, C. Masolo, A. Oltramari, L. Schneider, 2000, Sweetening Ontologies with DOLCE. Proceedings of EKAW 2002. Spain

[11] ISO 15926-2, 2003, ISO-15926:2003 Integration of lifecycle data for process plant including oil and gas production facilities: Part 2 – Data model.

[12] *Smith B.*, Against idiosyncrasy in Ontology Development. B. Bennett and C. Fellbaum (Eds.), Formal Ontology and Information Systems, (FOIS 2006)

[13] *Швецин Б.Я.* Онтология предприятия: экспириентологический подход. М.: ЛЕНАНД, 2010. 240 с.

[14] *Скобелев П.О.* Онтология деятельности для ситуационного управления предприятием [Текст] научный журнал «Онтология проектирования» №1, 2012, с. 7 – 38.

[15] *Виттих В.А.* Ситуационное управление с позиций постнеклассической науки / В.А. Виттих // Онтология проектирования. 2012. №2(3), с.7-15.

[16] *Повзун О.І.*, Козел Н.Б., Каратаєв О.А., Четвериков Г.Г. Концепції організації інформаційно-інтелектуальних технологій та інтелектуальної підтримки суспільно-економічних процесів: *k*-значні засоби. Частина 2// Бионика интеллекта. № 2(95), 2020, с. 51-62.

Поступила в редколлегию 10.11.2021

ПРАВИЛА
оформлення рукописів для авторів науково-технічного журналу «БІОНІКА ІНТЕЛЕКТУ»

Науково-технічний журнал «Біоніка інтелекту» приймає до друку написані спеціально для нього оригінальні рукописи, які раніше ніде не друкувались. Структура рукопису повинна бути такою: індекс УДК, відомості про авторів, заголовок, анотації (на трьох мовах), ключові слова, вступ, основний текст статті, висновки, список використаної літератури, резюме.

Відповідно до Постанови ВАК України від 15.01.2003 №7-05/1 (Бюлетень ВАК, №1, 2003, с. 2), стаття повинна мати такі необхідні елементи: постановка проблеми у загальному вигляді та її зв'язок із важливими науковими чи практичними завданнями; аналіз останніх досліджень і публікацій і виділення не вирішених раніше частин загальної проблеми в даній області; формулювання цілей та завдань дослідження; виклад основного матеріалу досліджень з повним обґрунтуванням отриманих наукових результатів; висновки з даного дослідження та перспективи подальших досліджень у даному напрямку.

Статті мають бути виконані в редакторі Microsoft Word. Формат сторінки – А4 (210×297 мм), поля: верхнє – 25 мм, нижнє – 20 мм, ліве, праве – 17 мм. Кількість колонок – 2, з інтервалом між ними 5 мм, основний шрифт Times New Roman, кегль основного тексту – 10 пунктів, міжрядковий інтервал – множник (1,1), абзацний відступ – 6 мм. Обсяг рукопису – від 6 до 12 сторінок (мови: українська, англійська, російська та мовою орігінала).

УДК друкується з першого рядка, без відступів, вирівнювання по лівому краю.

ПІБ автора (-ів), назва статті, назва та адреса учбового закладу необхідно надати повністю російською, українською та англійською мовами.

Назва статті друкується прописними літерами; шрифт прямий, напівжирний, кегль 12.

Назви розділів нумерують арабськими цифрами, виділяють жирним шрифтом. Відступи для назви статті, ініціалів та прізвищ авторів, відомостей про авторів, назв розділів, вступу та висновків, списку літератури: зверху – 6 пт, знизу – 3 пт.

Анотації (мовою статті, абзац 6–12 рядків, кегль 9) розміщують на початку статті, в ній має бути розміщена інформація про очікувані результати описаних досліджень (на трьох мовах).

Ключові слова (4–10 слів з тексту статті, які з точки зору інформаційного пошуку несуть змістовне навантаження) наводять мовою рукопису, через кому в називному відмінку, кегль 9.

Рисунки та *таблиці* (чорно-білі, контрастні) розміщуються у тексті після першого посилання у вигляді окремих об’єктів і нумерують арабськими цифрами наскрізною нумерацією за наявності більше ніж одного об’єкта. Невеликі схеми, що складаються з 3–4 елементів виконують, використовуючи вставку об’єкта Рисунок Microsoft Word. Більш складні виконують у графічних редакторах у вигляді чорно-білих графічних файлів форматів .tif, .jpg, .wmf, .cdr із розділенням 300 dpi. Рисунки мають міститися у текстовому файлі й обов’язково подаватися окремими файлами з відповідними назвами (наприклад, рис1.jpg).

Усі елементи рисунка, включаючи написи, повинні бути згруповані. Усі написи в рисунках і таблицях мають бути виконані шрифтом Times New Roman, кегль у рисунках – 10, у таблицях – 9.

Рисунок повинен мати центрований підпис (поза рисунком), шрифт 9, відступи зверху і знизу по 6 пт. Ширина рисунка має відповідати ширині колонки (або ширині сторінки).

Формули, символи, змінні повинні бути набрані в **редакторі формул MathType**. Формули розміщують посередині рядка й нумерують за наявності посилань на них у рукописі. Шрифт – Times New Roman. Висота змінної – 10 пунктів, великих і малих індексів – 8 пт, основний математичний символ – 12 (10) пт. Змінні, позначені латинськими літерами, набирають курсивом, грецькі літери, скорочення російських слів і цифри – прямим написанням. Змінні, які є в тексті, також набирають у редакторі формул.

Список літератури вміщує опубліковані джерела, на які є посилання в тексті, укладені у квадратні дужки, друкують без абзацного відступу, кегль 9 пт, відступ зверху – 6 пт.

Після списку літератури з відступом зверху 6 пт зазначають *дату подання статті до редколегії*. Число та місяць задають двозначними числами через крапку. Розмір шрифта – 9 пт, курсив, вирівнювання по правому краю.

Резюме (Times New Roman, кегль — 10 пунктів,) подають англійською мовою: обсяг резюме до 2000 знаків (бажаний переклад). *Структура резюме:* **Background, Materials and methods, Results, Conclusion.**

Разом із рукописом (на аркушах білого паперу формату А4 щільністю 80-90 г/м^2, надрукований на лазерному принтері) необхідно подати такі документи:

1. Заяву, яку повинні підписати всі автори.
2. Акт експертизи про можливість опублікування матеріалів у відкритому друці (якщо потрібно).
3. Рецензію, підписану доктором чи кандидатом наук.
4. Відомості про авторів.
5. Електронний варіант рукопису, резюме та відомостей про авторів.
6. Зробити оплату публікації.

Необхідно також зазначити один з наступних тематичних розділів, якому відповідає рукопис:

1. Теоретичні основи інформатики та кібернетики. Теорія інтелекту.
2. Математичне моделювання. Системний аналіз. Прийняття рішень.
3. Інтелектуальна обробка інформації. Розпізнавання образів.
4. Інформаційні технології та програмно-технічні комплекси.
5. Структурна, прикладна та математична лінгвістика.
6. Дискусійні повідомлення.

СОДЕРЖАНИЕ

НЕЙРОННЫЕ И ЛОГИЧЕСКИЕ СЕТИ

АЛГЕБРАИЗАЦИЯ ЛОГИКИ. РАСПОЗНАВАНИЕ И СИНТЕЗ РЕЧИ

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ. ОБЪЕКТНОЕ МОДЕЛИРОВАНИЕ

Наукове видання

БІОНІКА ІНТЕЛЕКТУ

інформація, мова, інтелект

Науково-технічний журнал

№ 2 (97)

2021

Головний редактор — *Г. Г. Четвериков*

Відповідальний редактор — *І.В. Кириченко*

Комп'ютерна верстка — *О. Б. Ісаєва*

Рекомендовано Вченою Радою
Харківського національного університету радіоелектроніки
(протокол № 11 от 24.12.2021)

Адреса редакції:

Україна, 61166, Харків-166, просп. Науки, 14,
Харківський національний університет радіоелектроніки, к. 127
тел. 702-14-77, факс 702-10-13,
e-mail: bionics@nure.ua

Підписано до друку 28.12.2021. Формат $60 \times 84\ ^1/_8$. Друк ризографічний.
Папір офсетний. Гарнітура Newton. Умов. друк. арк. 15,4. Обл.-вид. арк. 15,0.
Тираж 100 прим.

Віддруковано в редакційно-видавничому відділі ХНУРЕ
61166, Харків, просп. Науки, 14.