УДК 004.021 **DOI:** 10.18664/ikszt.voi6.151638

ДОЦЕНКО С. І., д. т. н., доцент,
РАДОУЦЬКИЙ К. Є., старший викладач,
ПАНАСЕНКО В. О., магістрант,
МАСТЮК А. С., магістрант,
КАМИНІНА Д. О., магістрант
(Український державний університет залізничного транспорту)

Огляд методів обробки та аналізу текстів на природних мовах

Розглянуто основні проблеми обробки природної мови, тобто необхідність структурування та систематизації даних, аналізу емоційного забарвлення тексту. Подано інформацію про принципи побудови глибоких мереж, рекурентних нейронних мереж, використання нейронних мереж з пам'яттю і машинного перекладу. Показано, що аналіз текстових даних має велике значення в сучасному світі, оскільки необхідно обробляти великі масиви текстової інформації.

Ключові слова: *обробка природної мови, аналіз тексту, обробка тексту, аналіз тональності тексту, класифікація, машинний переклад, нейронна мережа.*

Вступ

Під обробкою природних мов (Natural Language Processing, NLP) розуміють створення систем, що обробляють або "розуміють" мову з метою виконання певних завдань.

На сьогодні актуальність цього напрямку визначається необхідністю обробляти великі масиви текстової інформації, накопичені людством за останній час у глобальному інформаційному просторі. Тексти на природній мові – слабоструктурована інформація, тому їх обробка є непростим завданням, що виходить за рамки традиційної алгоритмічної обробки структурованих даних. Для того щоб отримати з текстів корисну інформацію, необхідно їх структурувати, упорядкувати, систематизувати, забезпечити пошук текстів за запитом користувача.

Аналіз останніх досліджень та публікацій

Наведемо основні класи задач, де використовуються методи аналізу текстів на природних мовах:

1) формування відповідей на питання (Question Answering) [1];

2) аналіз емоційного забарвлення висловлювань (Sentiment Analysis)

3) знаходження тексту, відповідного зображенню (Image to Text Mappings);

4) машинний переклад (Machine Translation) [2];

5) розпізнавання мови (Speech Recognition);

6) морфологічна розмітка (Part of Speech Tagging);

7) витяг сутностей (Name Entity Recognition).

У цих публікаціях розглянуто окремі методи аналізу текстів на природних мовах, але бракує аналізу застосування вказаних методів у конкретних галузях. Тому виникає задача проведення такого аналізу.

Мета і задачі дослідження

Мета цієї статті – виконати аналіз методів обробки та аналізу текстів на природних мовах та запропонувати методи обробки текстів для створення чатботів обслуговування клієнтів, машинного перекладу, навчання систем відповідей на питання глибоко розбиратися в неструктурованих або довгих текстах.

Далі розглянемо основні елементи, з яких можна будувати глибокі мережі для NLP.

Виклад основного матеріалу

Вектори слів

Кожне слово в цьому методі подається у вигляді d-вимірного вектора. Нехай d = 6. Вектор записується таким чином:

Uninterested = [_ _ _ _ _ _]

Наступним кроком є побудова матриці спільної зустрічальності (cooccurrence matrix). Розглянемо такий приклад виразу:

“I love NLP and I like dogs”

Матриця спільної зустрічальності містить кількість разів, скільки кожне слово зустрілося в корпусі (навчальному наборі) разом із кожним іншим словом цього корпусу. Результат зображено на рис. 1.

	I	Love	NLP	And	Like	Dogs
I	0	1	0	1	1	0
Love	1	0	1	0	0	0
NLP	0	1	0	1	0	0
And	1	0	1	0	0	0
Like	1	0	0	0	0	1
Dogs	0	0	0	0	1	0

Рис. 1. Матриця спільної зустрічальності

Рядки цієї матриці можуть служити як векторні подання наших слів, як зображено у векторах подання

$$\begin{aligned} I &= [010110] \\ Love &= [101000] \\ NLP &= [010100] \\ And &= [101000] \\ Like &= [100001] \\ Dogs &= [000010] \end{aligned} \quad (1)$$

Навіть з цієї простої матриці можливо почерпнути досить важливі відомості. Зауважимо, що вектори слів "love" та "like" містять одиниці в осередках, що відповідають за їх сусідство з іменниками ("NLP" і "dogs"). У них також стоїть "1" там, де вони є сусідами з "I", показуючи, що це слово, скоріш за все, дієслово. Простіше буде виявляти схожі риси, коли набір даних більший, ніж одна пропозиція: у цьому випадку вектори таких дієслів, як "love", "like" та інших синонімів, будуть схожі, так як ці слова будуть використовуватися у схожих контекстах.

Добре для початку, але розмірність вектора кожного слова буде лінійно зростати залежно від розміру корпусу. У разі мільйона слів (що небагато для стандартних завдань NLP), ми отримали б матрицю розмірності мільйон на мільйон, яка, до того ж, була б дуже розрідженою (з великою кількістю нулів).

Це, безумовно, не найкращий варіант з погляду ефективності зберігання даних. У питанні знаходження оптимального векторного подання слів було зроблено кілька серйозних зрушень. Найвідоміше з них – Word2Vec.

Word2Vec

Головна мета всіх методів ініціалізації вектора слова – зберігати в цьому векторі якомога більше інформації, дотримуючись розумної розмірності (в ідеалі, від 25 до 1000 слів). В основі Word2Vec лежить ідея навчитися прогнозувати навколишні слова для кожного слова. Розглянемо пропозицію з попереднього прикладу: "I love NLP and I like dogs". Зараз нас цікавлять тільки три перші слова. Нехай розмір нашого вікна дорівнює трьом, як у виразі

***"I love NLP** and I like dogs"*

Тепер візьмемо центральне слово "love" та передбачимо слова, що йдуть до і після нього за допомогою максимізації та оптимізації функції. Формально функція має максимізувати логарифмічну ймовірність кожного слова-контексту для поточного центрального слова [3].

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\le j\le m, j\neq 0}\log p(\omega_{t+j} \mid \omega_t). \quad (2)$$

З формули випливає, що логарифмічна ймовірність спільної зустрічальності додається як "I" і "love", так і "NLP" і "love" (в обох випадках "love" – центральне слово). Змінна T означає кількість навчальних (training) пропозицій. Розглянемо логарифмічну ймовірність детальніше у формулі

$$\log p(o \mid c) = \log \frac{\exp(u_o^T v_c)}{\sum_w^W \exp(u_w^T v_c)}, \quad (3)$$

де v_c – векторне подання центрального слова.

У кожного слова є два векторних подання: u_o і u_w, одне для випадку, коли слово займає центральну позицію, інше для випадку, коли це слово – "зовнішнє". Вектори навчаються методом стохастичного градієнтного спуску.

Word2Vec шукає векторні подання різних слів, максимізуючи логарифмічну ймовірність народження

слів контексту для даного центрального слова і перетворюючи вектори методом стохастичного градієнтного спуску [4].

Рекурентні нейронні мережі (Recurrent Neural Networks, RNN)

Головна перевага RNN у тому, що вони можуть ефективно використовувати дані з попередніх кроків. Маленький шматочок RNN зображено на рис. 2.

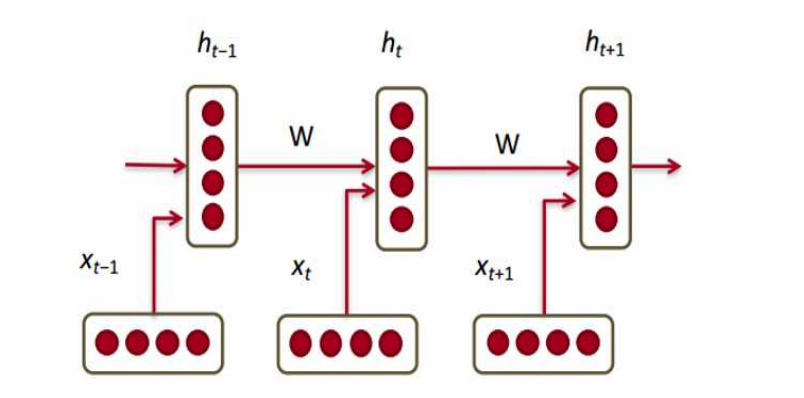

Рис. 2. Рекурентна нейронна мережа

Знизу зображені вектори слів (x_t, x_{t-1}, x_{t+1}). У кожного вектора на кожному кроці є прихований вектор стану (hidden state vector) (h_t, h_{t-1}, h_{t+1}). Будемо називати цю пару модулем (module) [5] . Модуль зображено на рис. 3.

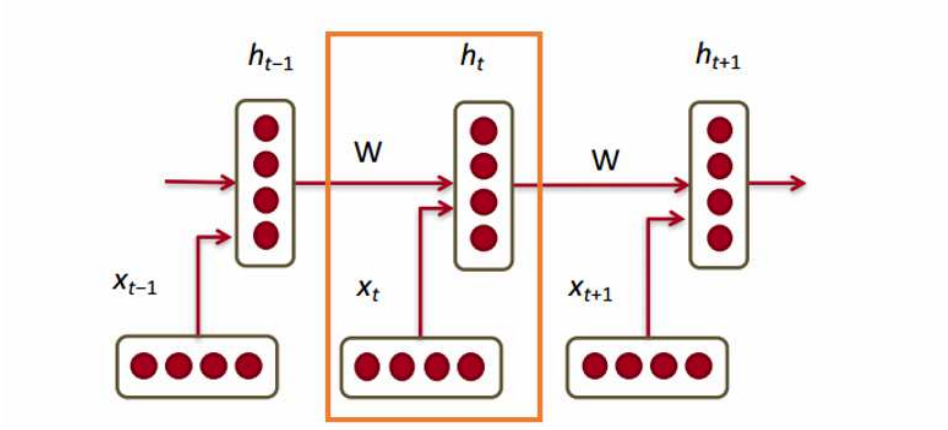

Рис. 3. Модуль рекурентної нейронної мережі

Прихований стан у кожному модулі RNN – це функція від вектора слова і вектора прихованого стану з минулого кроку, наведена у формулі

$$h_t = \sigma(W^{(hh)} h_{t-1} + W^{(hx)} x_{[t]}). \quad (4)$$

З формули випливає, що тут є матриця вагів W^{hx}, яку ми множимо на вхідне значення, і є рекурентна матриця вагів W^{hh}, яку множимо на вектор прихованого стану з попереднього кроку. Рекурентні матриці вагів на кожному кроці однакові. Це ключовий момент RNN. Цей підхід значно відрізняється від традиційних двошарових нейронних мереж. У цьому випадку зазвичай вибирається окрема матриця W для кожного шару: W_1 і W_2. Тут же рекурентна матриця вагів одна і та ж для всієї мережі.

Для отримання вихідних значень кожного модуля Y_{hat} служить ще одна матриця вагів – W^S, помножена на h. Вона зображена на рис. 4.

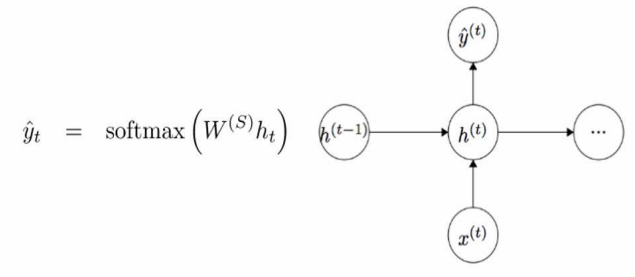

Рис. 4. Матриця вагів – W^S, помножена на h

Найбільш явна відміна RNN від традиційної нейронної мережі в тому, що RNN приймає на вхід послідовність вхідних даних (у нашому випадку слів). Цим вони відрізняються від типових CNN, на вхід яких подається ціле зображення. Для RNN вхідними даними може служити як коротке речення, так і твір з п'яти абзаців. Крім того, порядок, у якому подаються дані, може впливати на те, як у процесі навчання змінюються матриці вагів і вектори прихованих станів. До кінця навчання у векторах прихованих станів повинна накопичитися інформація з минулих кроків.

Керовані рекурентні нейрони (Gated recurrent units, GRU)

За допомогою керованого рекурентного нейрона проводиться обчислення векторів прихованих станів у RNN. Такий підхід дає змогу зберігати інформацію про більш віддалені залежності. Під час роботи методу зворотного поширення помилки (back propagation) помилка буде рухатися по RNN від останнього кроку до першого. При досить малому початковому градієнті (скажімо, менш 0,25) до третього або четвертого модуля градієнт майже зникне (так як за правилом похідної складної функції градієнти перемножуються), і тоді приховані стани перших кроків не оновляться.

У звичайних RNN вектор прихованих станів обчислюється за формулою

$$h_t = f(W^{(hh)}h_{t-1} + W^{(hx)}x_t). \quad (5)$$

Метод GRU дає змогу обчислювати h(t) інакше. Обчислення розбиваються на три блоки: фільтр поновлення (update gate), фільтр скидання стану (reset gate) і новий контейнер пам'яті (memory container) [6]. Фільтр-функції від вхідного векторного подання слова і прихованого стану на попередньому кроці наведені у формулах:

Update gate $$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1}), \quad (6)$$

Reset gate $$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1}). \quad (7)$$

Головна відмінність полягає в тому, що для кожного фільтра використовуються свої ваги. Це позначено різними верхніми індексами. Фільтр поновлення використовує W^z та U^z, а фільтр скидання стану – W^r та U^r.

Розрахунок контейнера пам'яті

$$\tilde{h}_t = \tanh(Wx_t + r_t \circ Uh_{t-1}). \quad (8)$$

Якщо множник фільтра скидання стану близький до нуля, то і весь твір також наблизиться до нуля, і, таким чином, інформація з попереднього кроку h_{t-1} не буде врахована. У цьому випадку нейрон – усього лише функція від нового вектора слова x_{t-1}.

Остаточна формула h(t)

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t, \quad (9)$$

де h_t – функція від усіх трьох компонентів: фільтра поновлення, фільтра скидання стану і контейнера пам'яті.

Розглянемо випадок, коли z_t наближається до 1 і коли z_t близький до 0.

У першому випадку вектор прихованого стану h_t у більшій мірі залежить від попереднього прихованого стану, а поточний контейнер пам'яті не береться до уваги, так як $(1 - z_t)$ прагне до 0. Коли ж z_t наближається до 1, новий вектор прихованого стану h_t, навпаки, залежить в основному від контейнера пам'яті, а попередній прихований стан не враховується. Три компонента формули описуються таким чином:

1) фільтр оновлення :

- якщо $z_t \sim 1$, то h_t не бере до уваги поточний вектор слова і просто копіює попередній прихований стан;
- якщо $z_t \sim 0$, то h_t не враховує попередній прихований стан і залежить тільки від контейнера пам'яті;
- цей фільтр дає змогу моделі контролювати, як багато інформації від попереднього прихованого стану повинно впливати на поточний прихований стан;

2) фільтр скидання стану:

- якщо $r_t \sim 1$, то контейнер пам'яті зберігає інформацію з попереднього прихованого стану;
- якщо $r_t \sim 0$, то контейнер пам'яті не враховує попередній прихований стан;
- цей фільтр дає змогу відкинути частину інформації, якщо в майбутньому вона не буде нас цікавити;

3) контейнер пам'яті залежить від фільтра скидання стану.

Нейронні мережі з пам'яттю (Memory Networks)

Нейронні мережі з пам'яттю використовуються для формування відповідей на питання. У публікаціях Джейсона Вестона (Jason Weston), Суміта Чопра (Sumit Chopra) і Антуана Бордес (Antoine Bordes) був уперше описаний клас моделей під назвою "мережі з пам'яттю".

Мережа з пам'яттю є унікальною, тому що у неї є асоціативна пам'ять, у яку вона може писати і з якої вона може читати. Таку пам'ять не використовують ні CNN, ні Q-Network (для навчання з підкріпленням (reinforcement learning), ні традиційні нейронні мережі. Це частково пов'язано з тим, що завдання формування відповідей на питання здебільшого покладається на здатність моделювати чи простежувати віддалені залежності, наприклад, стежити за героями історії або запам'ятовувати послідовність подій. У CNN або Q-Networks пам'ять як би вбудована у ваги системи, так як вона навчається від різних фільтрів або з допомогою карток відповідностей станів і дій. На перший погляд можна було б використовувати RNN або LSTM, але вони не здатні запам'ятовувати вхідні дані з минулого (що є критичним для задач формування відповідей на питання) [1].

Розглянемо, як така мережа обробляє вихідний текст. Як і більшість алгоритмів машинного навчання, перший крок – перетворити вхідні дані в подання в просторі ознак.

Наступний крок – взяти подання в просторі ознак I (x) і завантажити в пам'ять нову порцію вхідних даних x.

Пам'ять m подається як подоба масиву, складеного з окремих блоків пам'яті m_i. Кожен такий блок m_i може бути функцією від усієї пам'яті m, поданням у просторі ознак I(x) та / або самого себе. Функція G може просто зберігати все подання I(x) у блоці пам'яті m_i. Функцію G можна змінити так, щоб вона оновлювала пам'ять про минуле на основі нових вхідних даних. Третій і четвертий кроки містять у собі читання з пам'яті з урахуванням питання, щоб знайти подання ознак, і його декодування, щоб отримати остаточну відповідь.

Як опція R може служити RNN, що перетворює подання ознак у зрозумілі для людей і точні відповіді на питання.

Розглянемо крок 3. На цьому кроці порівняємо питання з кожним окремим блоком пам'яті й оцінимо, наскільки кожен блок підходить під відповідь на питання. Порівняємо питання за формулою

$$o_1 = O_1(x, m) = \arg\max_{i=1,\ldots,N} s_O(x, m_i). \qquad (10)$$

Щоб знайти подання найвідповіднішого питання, розраховується аргумент максимізації (argmax) оцінної функції. Оцінна функція обчислює матричний добуток між різними векторними поданнями питання й обраним блоком (або блоками) пам'яті. Вихідне подання потім передається RNN, LSTM або іншій оцінній функції, яка поверне зрозумілу для людини відповідь.

Навчання мережі проходить методом навчання з учителем, коли навчальні дані містять у собі вихідний текст, питання, що підтверджують пропозиції, і правильну відповідь. Цільова функція наведена у формулі

$$\sum_{\bar{f} \neq m_{01}} \max(0, \gamma - s_O(x, m_{01}) + s_O(x, \bar{f})) +$$
$$\sum_{\bar{f} \neq m_{02}} \max(0, \gamma - s_O([x, m_{01}], m_{02}) + s_O([x, m_{01}], \bar{f}')) +$$
$$\sum_{\bar{r} \neq r} \max(0, \gamma - s_R([x, m_{01}, m_{02}], r) + s_R([x, m_{01}, m_{02}], \bar{r})). \qquad (11)$$

Tree-LSTM для аналізу емоційного забарвлення висловлювань

Аналіз емоційного забарвлення – завдання визначення, що має висловлювати позитивну або негативну конотацію. Формально емоційне забарвлення можна визначити як "погляд на ситуацію або подію або ставлення до них". Найбільш поширеним інструментом для задач розпізнавання емоційного забарвлення є Tree-LSTM.

Ідея нелінійного розташування компонентів основана на думці, що природні мови демонструють властивість перетворювати послідовності слів у фрази. Ці фрази, залежно від порядку слів, можуть мати значення, звідки вони були видалені, значення входять до цих компонентів. Щоб відобразити цю властивість, мережа з декількох LSTM-нейронів має бути подана у вигляді дерева, де на кожен нейрон впливають його дочірні вузли.

Одна з відмінностей Tree-LSTM від звичайного LSTM полягає в тому, що в останньому прихований стан – функція від поточних вхідних даних та прихованого стану на попередньому кроці. У Tree-LSTM прихований стан – функція від поточних вхідних даних і прихованих станів його дочірніх нейронів. Схема зображена на рис. 5.

У Tree-LSTM кожен нейрон може містити в собі прихований стан усіх його дочірніх вузлів. Під час навчання мережа може усвідомити, що певне слово (наприклад, слово "не" або "дуже") надзвичайно важливе для визначення емоційного забарвлення всієї пропозиції. Можливість вище оцінити відповідний вузол забезпечує більшу гнучкість мережі і може покращити ефективність її роботи.

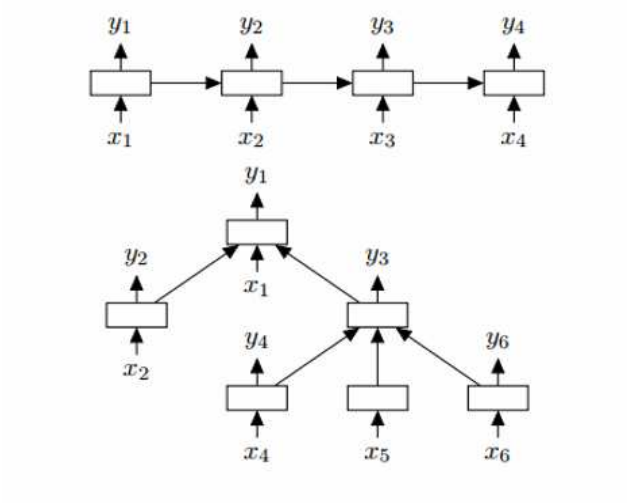

Рис. 5. Схема роботи Tree-LSTM

Нейронний машинний переклад (Neural Machine Translation, NMT)

Традиційні підходи до автоматичного перекладу передбачають знаходження пофразових відповідностей. Цей підхід вимагав відмінного знання лінгвістики і врешті-решт опинився недостатньо стабільним і нездатним до генералізації. Одна з проблем традиційного підходу полягала в тому, що вихідна пропозиція перекладалася по шматочках. Перекладати всю пропозицію за раз (як це робить NMT) більш ефективно, так як у цьому випадку залучається більш широкий контекст і порядок слів стає більш природним.

Новий підхід запропонували фахівці Google з машинного навчання Джефф Дін (Jeff Dean), Грег Коррадо (Greg Corrado), Оріал Віньялс (Orial Vinyals). Він описує глибоку мережу LSTM, яка може бути навчена за допомогою восьми шарів енкодерів і декодерів. Ми можемо розділити систему на три компоненти: енкодер RNN, декодер RNN і модуль "уваги" (attention module) [2]. Енкодер працює над завданням перетворення вхідної пропозиції у векторне подання, декодер повертає вихідне подання, потім модуль уваги повідомляє декодер, на чому слід акцентувати увагу під час операції декодування (тут вступає ідея використання всього контексту пропозиції). Схема роботи зображена на рис. 6.

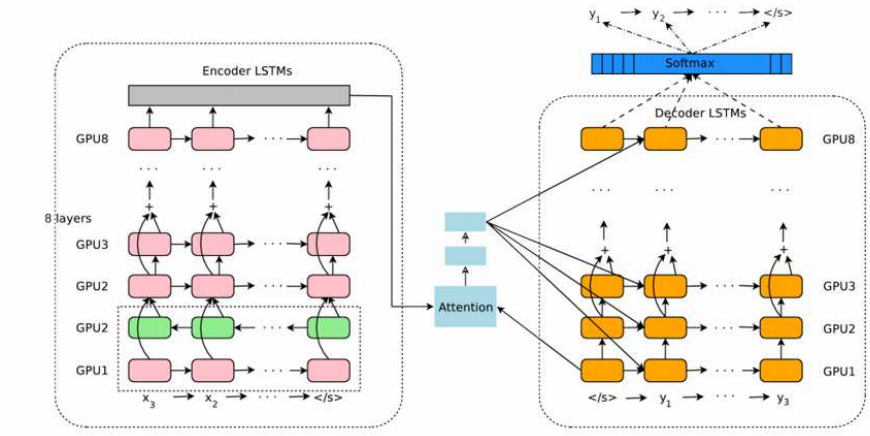

Рис. 6. Підхід фахівців Google з машинного навчання

Висновки

У статті були розглянуті методи обробки текстових даних. Існує велика кількість різноманітних методів обробки природних мов. Більшість методів обробки текстової інформації можуть використовуватись для більшості задач NLP, але існують такі методи, які застосовуються тільки для певних класів задач.

Підбиваючи підсумок, можна зробити висновок, що в сучасному світі, в середовищі постійно зростаючих обсягів інформації, аналіз текстових даних має великий потенціал і широке застосування.

Подальшими цілями в розвитку цієї галузі можуть бути поліпшення чатботів для обслуговування клієнтів, ідеальний машинний переклад і, можливо, навчання систем відповідей на питання глибоко розбиратися в неструктурованих або довгих текстах.

Список використаних джерел

1. Memory Networks [Electronic resource] : information/Notes for "Memory Networks" paper. – Available at: https://gist.github.com/shagunsodhani/c7a03a47b3d709e7c592fa7011b0f33e. (Accessed: 30.06.2018).
2. A Gentle Introduction to Neural Machine Translation [Electronic resource]: information/Neural Machine Translation – Available at: https://machinelearningmastery.com/introduction-neural-machine-translation. (Accessed: 20.07.2018).

3. Word2Vec. [Electronic resource]: information/Word2VecIntroduction. –Available at: https://skymind.ai/wiki/word2vec. (Accessed: 20.06.2018).
4. DeepDiveTutorial. Extracting mentions of spouses from the news [Electronic resource]: information/ DeepDive. – Available at: http://deepdive.stanford.edu/example-spouse. - (Accessed: 07.06.2018).
5. Zhang, X. Character-level convolutional networks for text classification [Electronic resource]: informa3. Zhang, X. Character-level convolutional networks for text classification [Electronic resource]: information/In Advancesin Neural Information Processing Systems. – Available at:http://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification.pdf. (Accessed: 29.06.2018).
6. Andrej Karpathy The Unreasonable Effectiveness of Recurrent Neural Networks [Electronic resource]: information/RecurrentNeuralNetworks – Available at: http://karpathy.github.io/2015/05/21/rnn-effectiveness. (Accessed: 09.07.2018).

Доценко С. И., Радоуцкий К. Е., Панасенко В. О., Мастюк А. С., Камынина Д. А. Методы обработки и анализа текстов на природных языках.

Аннотация. В данной статье рассматриваются основные проблемы обработки естественного языка, то есть необходимость структурирования и систематизации данных, анализа эмоциональной окраски текста. В статье представлена информация о принципах построения глубоких сетей, рекуррентных нейронных сетей, использовании нейронных сетей с памятью и машинного перевода. Показано, что анализ текстовых данных имеет большое значение в современном мире, поскольку необходимо обрабатывать большие массивы текстовой информации.

Ключевые слова: обработка естественного языка, анализ текста, обработка текста, анализ тональности текста, классификация, машинный перевод, нейронная сеть.

Dotsenko S., Radoutsky K., Panasenko V., Mastiuk A., Kamynina D. Methods of text`s processing and analysis in natural languages.

Abstract. The article analyzes the methods of processing and analysis of texts in natural languages and discusses the main problems of natural language processing, that is, the necessity of data structuring and data systematization. The analysis of emotional coloring of the text is also important. These include the principles of the construction and operation of deep networks and recurrent neural networks. The article provides information about the using of neural networks with memory to form answers custom questions. It describes the use of Tree-LSTM used to analyze the emotional coloring of statements. The use of neural machine translation to find phrasal matches is also considered. The relevance of this direction is determined by the need to process large amounts of textual information which were accumulated by mankind lately in the global information space. It is shown that the analysis of text data has a great importance in the modern world.

Key words: natural language processing, text analysis, text processing, text tonality analysis, classification, machine translation, neural network.

Надійшла …02.11.2018 р.

***Доценко Сергій Ілліч**, д. т. н., доцент кафедри «Спеціалізовані комп'ютерні системи». Український державний університет залізничного транспорту. E-mail: docenko@kart.edu.ua. ORCID: http://orcid.org/0000-0003-3021-4192.*

***Радоуцький Костянтин Євгеньович**, старший викладач кафедри «Спеціалізовані комп'ютерні системи». Український державний університет залізничного транспорту. E-mail: 7x24@gmail.com. ORCID: http://orcid.org/0000-0003-3610-535X.*

***Панасенко Владислав Олегович**, магістрант кафедри «Спеціалізовані комп'ютерні системи». Український державний університет залізничного транспорту. E-mail: panasenko_vladislav@protonmail.com.*

***Мастюк Антон Сергійович**, магістрант кафедри «Спеціалізовані комп'ютерні системи». Український державний університет залізничного транспорту. E-mail: anton.mastiuk@gmail.com.*

***Каминіна Дар'я Олександрівна**, магістрант кафедри «Спеціалізовані комп'ютерні системи». Український державний університет залізничного транспорту. E-mail: dkamdash@gmail.com.*

***Dotsenko Sergei**, Doctor of Technical Sciences, assistant professor, Department of Specialized Computer Systems, Ukrainian state university of railway transport. E-mail: docenko@kart.edu.ua. ORCID: http://orcid.org/0000-0003-3021-4192.*

***Radoutsky Konstantin**, Senior Lecturer, Department of Specialized Computer Systems, Ukrainian state university of railway transport. E-mail: 7x24@gmail.com. ORCID: http://orcid.org/0000-0003-3610-535X.*

***Panasenko Vladyslav**, master student, Department of Specialized Computer Systems, Ukrainian state university of railway transport. E-mail: panasenko_vladislav@protonmail.com.*

***Mastiuk Anton**, master student, Department of Specialized Computer Systems, Ukrainian state university of railway transport. E-mail: anton.mastiuk@gmail.com.*

***Kamynina Daria**, master student, Department of Specialized Computer Systems, Ukrainian state university of railway transport. E-mail: dkamdash@gmail.com.*