УДК 004.83

С. Д. Погорілий, А. А. Крамов
Київський національний університет імені Тараса Шевченка
Проспект Академіка Глушкова, 4Г, 03022 Київ, Україна

Метод розрахунку когерентності українського тексту

Проаналізовано основні методи автоматизованої оцінки когерентності тексту, який написано природною мовою. Запропоновано вдосконалення методу графа семантичної схожості за допомогою попередньої підготовки моделі, а саме здійснення навчання нейронної мережі векторного представлення речень. Проведено експериментальну перевірку роботи методу графа семантичної схожості та його модифікованих версій на множині україномовних статей наукових журналів різної тематики. Ефективність роботи методу та його модифікацій розраховано за допомогою вирішення типових задач оцінки когерентності тексту: задач розрізнення документів і вставки. На основі отриманих результатів визначено найбільш ефективні модифікацію та параметри методу графа семантичної схожості для оцінки когерентності україномовних текстів.

Ключові слова: *обробка природної мови, когерентність тексту, граф семантичної схожості, нейронна мережа, модель Doc2Vec, семантична міра схожості речень.*

Вступ

У зв'язку з постійним розвитком технологій машинного навчання та збільшенням обчислювальної потужності робочих станцій, спостерігається зростаючий інтерес науковців до вирішення низки задач, що неформально розглядаються як *AI-повні*. Під AI-повними задачами розуміють проблеми, для вирішення яких необхідно створити систему, що повністю відтворює розум людини [1]. Особлива увага приділяється *обробці природної мови* (англ. *Natural language processing — NLP*), про що свідчить наявність актуальних робіт і конференцій [2], які пов'язані з NLP. До задач NLP відноситься широкий спектр завдань: синтез мовлення, розпізнавання мови, машинний переклад, інформаційний пошук, спрощення тексту тощо. Задачі NLP актуальні не тільки для галузі комп'ютерної лінгвістики, але й для інших областей науки: метеорології, медицини, геофізики тощо. Активним дослідженням задач природної обробки тексту займаються науковці Стенфордського університету та корпорації Google [2, 3]. Українськими науковцями теж здійснюється дослідження автоматизованої обробки текстів української мови, про що

свідчить наявність корпусів, частотних словників, тезаурусів і програмних моделей обробки природної української мови [4].

Враховуючи актуальність пошукової оптимізації сайтів у мережі Інтернет, доцільним стає автоматизоване здійснення оцінки якості написаного тексту для покращення розуміння матеріалу читачем і підвищення позиції статті відносно конкуруючих веб-сторінок у видачі пошукової системи. До складових оцінки якості тексту відноситься поняття *когерентності*. Під когерентністю тексту розуміють цілісність тексту, що полягає в логіко-семантичній, граматичній і стилістичній співвіднесеності та взаємозалежності його складових [5]. Лінгвістами здійснюється структурний поділ поняття когерентності на три види: локальну, глобальну та тематичну. *Локальна когерентність* визначається міжфразовими синтаксичними зв'язками; *глобальна* — забезпечує єдність тексту як смислового цілого, його внутрішню цілісність. *Тематична когерентність* проявляється за допомогою ключових слів, які тематично поєднують фрагменти тексту. В цілому, когерентність тексту дозволяє оцінити тематичну узгодженість складових частин тексту, його смислову цілісність і, відповідно, зрозумілість матеріалу тексту читачу. Питання оцінки когерентності тексту активно розглядається науковцями: запропоновано низку методів для знаходження міри цілісності тексту за допомогою машинних методів навчання. Однак основний вектор актуальних досліджень напрямлений на оцінку когерентності текстів англійської мови; для української мови, що має свої морфологічні, синтаксичні та семантичні особливості, необхідно здійснювати індивідуальне дослідження. Тому актуальною є задача оцінки когерентності текстів української мови за допомогою існуючих методів, а також їхнє подальше вдосконалення відповідно до особливостей мови.

Метою цієї роботи є порівняльний аналіз існуючих методів оцінки когерентності тексту природної мови; вдосконалення певного методу та здійснення експериментальної перевірки ефективності роботи різних модифікацій обраного методу для корпусу української мови.

Огляд існуючих методів оцінки когерентності тексту

Вхідними даними кожного методу є текст природної мови. Результатом роботи є числове значення — міра когерентності тексту. Розглянемо основні методи оцінки цілісності тексту, після чого зупинимося на більш детальному огляді певного методу.

У 2008 році була запропонована модель оцінки когерентності тексту, яка отримала назву *Entity Grid* [6]. Головна ідея моделі полягає в припущенні, що розподіл ключових сутностей тексту (іменні групи, які присутні в реченнях) підпорядковується певній закономірності. Параметром оцінки когерентності вважається частота зміни ролі ключових сутностей у тексті (підмет, прямий додаток, інше), тобто аналізується частота зміни наголосів у тексті, які привертають увагу читача. У випадку різких/рівномірних переходів від однієї ключової сутності до іншої оцінка когерентності відповідно зменшується/збільшується. Для навчання моделі автори використовували метод опорних векторів (support vector machine — SVM). Таким чином, результат методу Entity Grid є дискретним значенням: 1 — якщо текст когерентний, 0 — в іншому випадку.

У 2013 році був запропонований новий метод оцінки когерентності тексту, який отримав назву *Entity Graph* [7]. У методі Entity Graph здійснюється побудова графа, що інтерпретує текст, для оцінки взаємозв'язку між сусідніми та віддаленими реченнями (на відміну від Entity Grid, де переважно розглядався локальний зв'язок речень). Текст представлений за допомогою орієнтованого двочасткового графа (біграфа). Перша підмножина вершин графа відповідає набору речень тексту. У другій підмножині графа кожній вершині ставиться у відповідність сутність. Зважене ребро між вершинами речення та сутності встановлюється у разі наявності сутності в реченні. Вага ребра встановлюється залежно від ролі сутності в реченні (підмет, прямий додаток, без ролі). Отриманий орієнтований двочастковий граф далі перетворюється в орієнтований проекційний граф, в якому всі вершини позначають речення тексту, а ребра між ними встановлюються за умови наявності щонайменше однієї спільної сутності. Напрямок ребер відповідає послідовності розміщення речень у тексті. Ваги ребер розраховуються відповідно до трьох різних проекційних методів. Для оцінки когерентності тексту обраховується середнє арифметичне значення напівстепені виходу вершин графа.

Існують методи, що використовують різнотипні *нейронні мережі* для оцінки когерентності тексту. Відповідні методи реалізовані на основі рекурсивної і рекурентної [8] і згорткової нейронних мереж [9]. Основна ідея полягає в початковому розбитті тексту на речення та подальшому перетворенні «речення» – «вектор»; таку функцію виконують початкові шари нейронних мереж. Варто зазначити, що для такого перетворення використовуються заздалегідь навчені моделі векторного представлення слів: Word2Vec, GloVe та їхньої модифікації. Після здійснення перетворення «речення» – «вектор» наступними шарами нейронних мереж виконується пошук закономірностей між векторами речень згідно зі структурою мережі та значеннями ваг ребер. Вихідним результатом є числове значення; чим більше числове значення, тим вища оцінка когерентності. Недоліком є відсутність нормалізації вихідного значення та непрозорість моделі: неможливо відслідкувати причину встановлення низької/високої оцінки когерентності вхідного тексту.

Останнім розглянемо метод *графа семантичної схожості* (ГСС) [10]. Як і Entity Graph, ГСС відноситься до графічних методів оцінки когерентності тексту. Спочатку текст трансформується до орієнтованого графа $G(V,E)$, де V — множина вершин, E — множина ребер графа. Вершина $v_i \in V$ відповідає i-му реченню s_i в тексті, а зважене напрямлене ребро $e_{ij} \in E$ описує міру семантичної схожості між реченнями s_i та s_j. Представлення речення здійснюється у векторному вигляді. Для цього кожне слово речення трансформується у вектор за допомогою попередньо навченої моделі Word2Vec чи GloVe. Таким чином, речення складається з множини слів $\{w_1, w_2, ..., w_M\}$ (M — кількість слів речення), де кожному слову відповідає вектор $\{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_M\}$. Речення s може бути представлене як вектор $\mathbf{s}$ за допомогою усереднення векторів його слів:

$$\mathbf{s} = \frac{1}{M}\sum_{k=1}^{M}\mathbf{w}_k \ . \qquad (1)$$

Для формування графа пропонується використовувати три різних підходи: PAV, SSV, MSV.

У підході PAV (preceding adjacent vertex — попередня сусідня вершина) здійснюється формування ребер від вершини кожного речення до вершини сусіднього попереднього речення, якщо міра схожості між цими реченнями більша 0; в іншому випадку здійснюється спроба встановити ребро між поточною та іншою найближчою попередньою вершинами. Міра схожості речень розраховується за наступною формулою:

$$\mathrm{sim}(s_i, s_j) = \alpha \,\mathrm{uot}(s_i, s_j) + (1-\alpha)\cos(\mathbf{s}_i, \mathbf{s}_j), \qquad (2)$$

де uot — відношення кількості спільних сутностей речень s_i та s_j до загальної кількості сутностей цих речень; $\cos(\mathbf{s}_i, \mathbf{s}_j)$ — косинусна відстань між векторами речень; α — регулятивний параметр, $\alpha \in [0,1]$.

На відміну від PAV, у підході SSV (single similar vertex — єдина схожа вершина) залежність між реченнями може мати двонаправлений характер, тому ребра можуть бути направлені не тільки на вершини попередніх речень. Однак зберігається обмеження значення напівстепені виходу кожної вершини, не більшого за 1.

Для встановлення вихідного ребра поточної вершини здійснюється пошук речення, яке є найбільш схожим до поточного у семантичному розумінні. Міра схожості, відповідна вазі ребра, розраховується як відношення косинусної відстані між векторами речень до відстані між реченнями в тексті:

$$\mathrm{weight}(e_{ij}) = \frac{\cos(\mathbf{s}_i, \mathbf{s}_j)}{|i-j|}. \qquad (3)$$

У попередніх підходах PAV і SSV кожна вершина була або інцидентна одному ребру, що направлене до іншої вершини, або ізольованою. У підході MSV (multiple similar vertex — багато схожих вершин) це обмеження ігнорується: для кожного речення здійснюється пошук усіх речень, міра схожості (див. формулу (3)) яких з поточним реченням більша за порогове значення θ. Ваги сформованих ребер рівні мірі схожості суміжних вершин. Приклади всіх підходів до формування ГСС зображено на рис. 1.

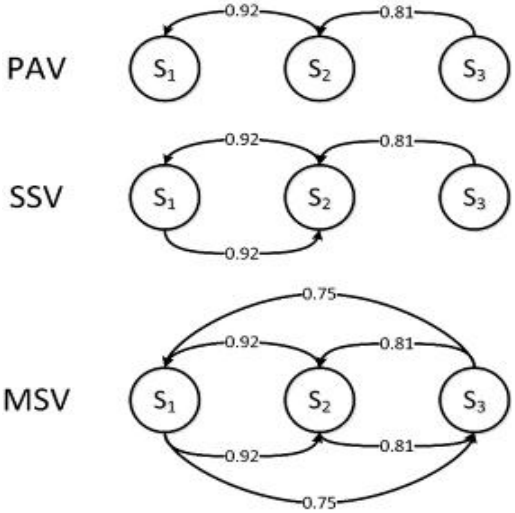

Рис. 1. Приклади підходів до побудови ГСС

Оцінка когерентності тексту t_c здійснюється за рахунок аналізу сформованого графа і обраховується наступним чином:

$$t_c = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{L_i}\sum_{k=1}^{L_i}\mathrm{weight}(e_{ik}), \qquad (4)$$

де N — кількість речень у тексті; L_i — кількість ребер, які виходять з вершини v_i. Значення t_c нормалізоване (приймає значення з діапазону $[0,1]$) і прямо пропорційне мірі когерентності тексту.

Графічні методи оцінки когерентності тексту (ГСС, Entity Graph) дозволяють водночас відстежити процес формування вихідного результату методу, а також проаналізувати тематичну схожість певних речень для подальшого покращення цілісності тексту. Моделі інших методів розглядаються як «чорний ящик», що унеможливлює аналіз причини формування вихідної оцінки когерентності тексту та виконання її підвищення. Особливістю методу ГСС, порівняно з Entity Graph, є аналіз власне семантичної складової тексту: Entity Graph використовує синтаксисний аналіз іменованих сутностей для відстеження зміни їхніх ролей. Також до переваг ГСС варто віднести неперервність і нормалізацію вихідної оцінки когерентності тексту. Таким чином, проаналізувавши основні методи оцінки когерентності тексту, було вирішено використати метод ГСС для здійснення експериментальної перевірки ефективності роботи методів на корпусі української мови.

Модифікація методу ГСС

Принцип роботи методу ГСС полягає в побудові орієнтованого зваженого графа, вершини якого представляють речення тексту, а ваги ребер — семантичну схожість речень. Когерентність оцінюється як середнє арифметичне значення ваг ребер графа. Нетривіальною виглядає задача знаходження міри семантичної схожості речень, яка розраховується відповідно до обраного підходу (PAV, SSV, MSV) за допомогою попереднього розрахунку векторного представлення речень. Вектор кожного речення обчислюється як середнє арифметичне векторів слів, що входять до речення (див. формулу (1)). Для векторного представлення слів попередньо виконується навчання нейронної мережі Word2Vec або GloVe.

Такий підхід до векторного представлення речень використовувався в багатьох роботах, які пов'язані з машинною обробкою тексту, та виявився доволі ефективним. Однак описаний підхід має декілька недоліків, а саме:

— неможливо здійснювати векторне представлення слова, незнайомого навченій моделі, без додаткового етапу навчання;

— не вирішується проблема омонімії, адже кожне слово має єдине векторне представлення незалежно від контексту;

— не враховується порядок слів у тексті.

Відсутність слова у словнику навченої моделі обумовлює або ігнорування слова в тексті, або пошук найбільш схожого слова. Зрозуміло, що така апроксимація погіршує семантичне представлення речення в цілому. Ігнорування порядку слів у тексті може призвести до некоректного смислового навантаження речень (варто зазначити, що для різних мов фактор впливу порядку слів у реченні відрізняється). Крім того, векторне представлення слова не вирішує проблему омонімії, адже словам, які однаково звучать і пишуться, але мають різне значення, відповідатиме спільний вектор; таким чином, не враховується контекст вживання слова в реченні.

Вирішити описані вище проблеми пропонується за допомогою навчання нейронної мережі векторного представлення документів — Doc2Vec [11]. Навче-

на модель Doc2Vec дозволяє здійснювати векторне представлення групи слів, речень, абзаців тощо (залежно від конкретної задачі). У випадку знаходження семантичної схожості речень доцільно здійснювати навчання моделі на колекціях речень. Процес навчання нейронної мережі може відбуватися за допомогою підходу «навчання без учителя», що дозволяє використовувати нерозмічені дані та виконувати векторне представлення речень, незнайомих навченій моделі. В роботі [11] описано дві моделі Doc2Vec: Distributed Bag of Words (DBOW) і Distributed Memory (DM). Відмінність моделей полягає в наступному: модель DM, на відміну від DBOW, при навчанні враховує порядок слів речення. Це зумовлює більше використання пам'яті моделлю DM і повільнішу роботу порівняно з методом DBOW. Можливим є здійснення об'єднання моделей DBOW і DM (далі DBOW+DM), хоча таку комбінацію моделей методу Doc2Vec варто розглядати як експериментальну.

Експериментальна перевірка роботи методу ГСС

Експериментальну перевірку роботи методу ГСС варто розділити на наступні етапи:

— генерацію навчального корпусу та тестової вибірки;

— навчання моделей нейронних мереж;

— програмну реалізацію різнотипних моделей ГСС;

— тестування ефективності моделей у вирішенні типових задач оцінки когерентності тексту.

Генерація навчального корпусу та тестової вибірки

Першим етапом експериментальної перевірки роботи методу ГСС стало формування навчальної вибірки для нейронних мереж Word2Vec, DBOW і DM. Унаслідок пошуку існуючих корпусів української мови було знайдено готові рішення, що створені відкритою спільнотою фахівців lang-uk [4]. Представники спільноти lang-uk надали відкритий доступ до попередньо оброблених корпусів української мови, сформованих з новин, творів художньої літератури, статей ресурсу «Вікіпедія». Однак тексти, що розміщені в корпусах, представлені у вигляді перемішаних речень згідно з вимогами правовласників текстів. Для тестування методу ГСС необхідні оригінальні версії текстів, тому було прийнято рішення сформувати навчальну вибірку власноруч. Як вхідні дані було вирішено обрати публікації українських наукових журналів. На сайтах наукових журналів повний текст публікацій доступний лише в PDF-форматі, що значно ускладнює процес формування навчальної вибірки, адже необхідні додаткові операції екстракції власне тексту статті в належному форматі. Тому для формування навчального корпусу було вирішено використати анотації статей, які можливо отримати з HTML-розмітки. Враховуючи, що анотації містять основні тематичні терміни, які стосуються відповідних статей, такий підхід повинен дозволити водночас навчити моделі працювати з реченнями, які стосуються певної тематики, а також зменшити об'єм навчальної вибірки. Для перевірки цього припущення як тестовий набір обрано повні версії статей, екстраговані з PDF-файлів. Генерацію навчального корпусу та тестового набору даних виконано окремим консольним застосуванням, написаним мовою програмування Python 3.6. Збір HTML-сторінок і PDF-файлів виконано за допомогою

реалізації пошукового робота, який працює як HTTP-клієнт [12, 13]. Було здійснено програмний обхід 266 веб-сайтів українських наукових журналів, які відносяться до категорій «Природничі та точні науки» та «Соціогуманітарні науки». Значна кількість статей, які розміщені на проаналізованих веб-ресурсах, написана російською або англійською мовами, тому додатково було виконано автоматизоване детектування української мови програмним пакетом *langdetect*, розробленим корпорацією Google.

Наступним кроком генерації навчального корпусу було виконання попередньої обробки текстів, а саме: 1) розбиття тексту на речення; 2) вилучення стоп-слів; 3) здійснення лематизації слів.

Вилучення стоп-слів з речень виконане за допомогою ітераційної перевірки наявності слова в списку стоп-слів української мови. Лематизація — це процес перетворення слова у словниковий вид або лему. Іменники та прикметники лематизація трансформує у форму однини, дієслово — в інфінітив (відповідь на питання «що робити?»). На жаль, для мови програмування Python відсутні пакети лематизації української мови. Єдиним рішенням, знайденим для української мови, стали утиліти, що створені попередньо зазначеною відкритою спільнотою фахівців lang-uk. Утиліти написані мовою програмування Groovy, тому було створено «обгортку» мовою програмування Python для виконання утиліт у паралельному потоці програми. За допомогою утиліт було здійснено водночас розбиття тексту на речення та лематизація слів. Для створення навчальної вибірки було використано 74 180 анотацій; загальна кількість речень для навчання моделі становить 355 537. Для формування тестового набору текстів необхідно було виконати екстракцію матеріалу статті із PDF-файлу. Існуючих рішень для українських наукових публікацій не було знайдено у відкритому доступі, тому було вирішено використати інше рішення — клієнт-серверне застосування Science Parse [14]. У результаті було сформовано близько 1000 тестових текстів.

Навчання моделей нейронних мереж

Навчання нейронних моделей Word2Vec, DBOW і DM на сформованій навчальній вибірці було виконано за допомогою окремого застосування, написаного мовою програмування Python 3.6. Для виконання навчання було використано готові класи *Word2Vec* і *Doc2Vec* пакету *genism* [15]. Для прискорення процесу навчання було використано розширення Cython. Навчання здійснювалося з наступними параметрами:

1) розмірність вектора — 300;
2) кількість епох — 50;
3) розмір «вікна» — 10;
4) порогове значення частотного словника — 1.

Запуск застосування було здійснено на робочій станції з наступними характеристиками: Intel Core i7-7700 (3,6–4,2 ГГц) / RAM 32 ГБ / SSD. Результати навчання нейронних мереж — моделі Word2Vec, DBOW, DM — збережено на файловій системі з можливістю їхнього завантаження в пам’ять програми та подальшого використання.

Програмна реалізація різнотипних моделей ГСС

Програмна реалізація ГСС здійснена мовою програмування Python 3.6. Застосування містить 3 базових класи, кожен з яких реалізує відповідний підхід до побудови ГСС: PAV, SSV, MSV. Базові класи наслідують батьківський клас, в якому описані загальні функції для дочірніх класів. Для кожного класу передбачено можливість змінити модель (для здійснення векторного представлення речень) і функцію обрахунку міри семантичної схожості речень. Для графів PAV і MSV спроектовано можливість змінювати параметри налаштування: регулятивний параметр α для PAV і порогове значення θ для MSV. Результат роботи вказаних підходів залежить від обраних значень параметрів, що потребує подальшої експериментальної перевірки.

Тестування

Задача розрізнення документів (англ. document discrimination task — DDT) полягає в наступному: для документа з перевірочної множини обраховується його когерентність; потім речення тексту випадковим чином перемішуються, і обраховується когерентність зміненого тексту. Якщо значення когерентності оригіналу більше значення модифікованого тексту, вважається, що текст оброблений вірно. Оцінка задачі розрізнення документів S_{DDT} розраховується наступним чином:

$$S_{DDT} = \frac{N_{correct}}{N_{total}}, \tag{5}$$

де $N_{correct}$ — кількість вірно оброблених документів; N_{total} — загальна кількість документів.

Для ГСС PAV задача розрізнення документів виконувалася для різних значень регулятивного параметра α з кроком 0,1. Аналогічно здійснювалося тестування для порогового значення θ ГСС MSV. У таблиці нижче наведені максимальні результати вирішення задачі розрізнення документів методом ГСС з відповідними підходами та їхні параметрами для тестової множини документів. На відміну від підходу SSV, результати роботи PAV і MSV залежать від параметрів налаштування. Проаналізуємо більш детально залежність ефективності зазначених підходів від зміни параметра. На рис. 2 зображено динаміку зміни оцінки задачі розрізнення документів відносно регулятивного параметра α для підходу PAV. На графіку спостерігається поступове підвищення точності роботи підходу PAV зі збільшенням регулятивного параметра; така залежність вказує на необхідність урахування різноманітних елементів міжфразової єдності (синонімів, антонімів, кореферентів) при здійсненні оцінки когерентності текстів української мови. Щодо порівняльної характеристики моделей, різні варіанти Doc2Vec виявились ефективнішими, порівняно з моделлю Word2Vec. Найвищі результати отримані за допомогою конкатенації моделей DBOW+DM (проігноровано значення моделі Word2Vec при $\alpha = 0$ у зв'язку зі значним відхиленням від множини інших отриманих значень моделі), однак необхідно зазначити наявність непропорційних змін вихідних результатів відносно регулятивного параметра в моделях DBOW і DBOW+DM. На рис. 3 зображено графік зміни результату задачі розрізнення до-

кументів відносно порогового значення θ для підходу MSV. На графіку простежується зменшення ефективності роботи кожної моделі при збільшенні порогового значення. Така залежність вказує на необхідність урахування зв'язку між всіма реченнями тексту, адже максимальний результат спостерігається при пороговому значенні $\theta = 0$. Найкращий результат вирішення задачі розрізнення документів отриманий за допомогою моделі DM. Незважаючи на високу ефективність комбінованої моделі DBOW+DM, порівняно з іншими моделями, для підходів PAV і SSV, її результати роботи для підходу MSV виявилися найнижчими.

Максимальні результати вирішення задач розрізнення документів і вставки методом ГСС

Підхід	Параметр	Задача розрізнення документів		Задача вставки	
		Модель	Значення	Модель	Значення
PAV	$\alpha = 0,8$	DBOW	0,661	DM	0,532
SSV	–	DBOW+DM	0,628	DBOW+DM	0,227
MSV	$\theta = 0$	DM	0,808	DM	0,76

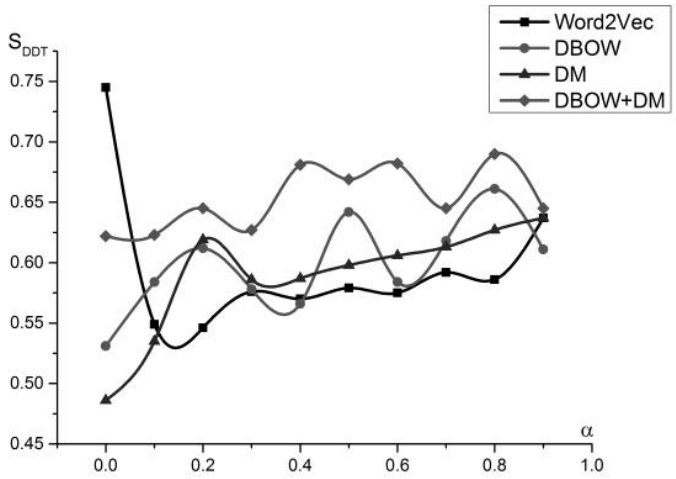

Рис. 2. Динаміка зміни оцінки задачі розрізнення документів відносно регулятивного параметра α для підходу PAV

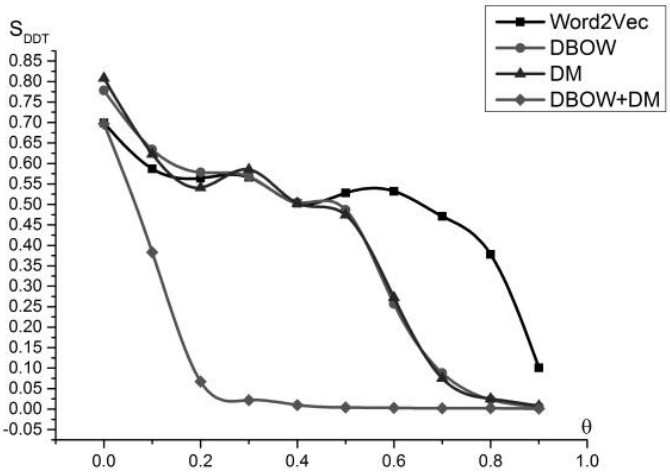

Рис. 3. Динаміка зміни оцінки задачі розрізнення документів відносно порогового значення θ для підходу MSV

Задача вставки. Алгоритм дій для виконання задачі вставки (англ. insertion task — IT) наступний: з тексту випадковим чином вилучається одне речення; потім видалене речення вставляється почергово у всі можливі позиції тексту. Для кожного варіанта вставки обраховується когерентність тексту. Обирається варіант вставки з найбільшим значенням когерентності. Якщо обраний варіант вставки співпадає з оригінальним текстом, тоді задача вставки вважається виконаною вірно для цього тексту. Оцінкою задачі вставки є значення S_{IT}, що розраховується за наступною формулою:

$$S_{IT} = \frac{N_{correct}}{N_{total}}, \tag{6}$$

де $N_{correct}$ — кількість вірно оброблених документів; N_{total} — загальна кількість документів. У таблиці наведені максимальні результати вирішення задачі вставки методом ГСС для тестової множини документів. Розглянемо детальніше залежності підходів PAV і MSV від параметрів налаштування. На рис. 4 зображено графік залежності оцінки задачі вставки від значення регулятивного параметра α для підходу PAV. Порівняно з моделлю Word2Vec, всі інші моделі Doc2Vec виявились ефективнішими. Найкращий результат було отримано для моделі DM. Аналогічно до задачі розрізнення документів, на графіку спостерігається покращення ефективності роботи підходу PAV зі збільшенням регулятивного параметра α.

На рис. 5 зображено графік зміни оцінки задачі вставки відносно порогового значення θ для підходу MSV. Аналогічно до задачі розрізнення документів найбільш ефективною виявилася модель DM, а комбінація моделей DBOW+DM показала найнижчі результати. Результати, що отримані для задачі вставки за допомогою підходу MSV, підтверджують необхідність розгляду вхідного тексту як насиченого графа: кожна вершина, що представляє речення, суміжна з будь-якою іншою вершиною графа.

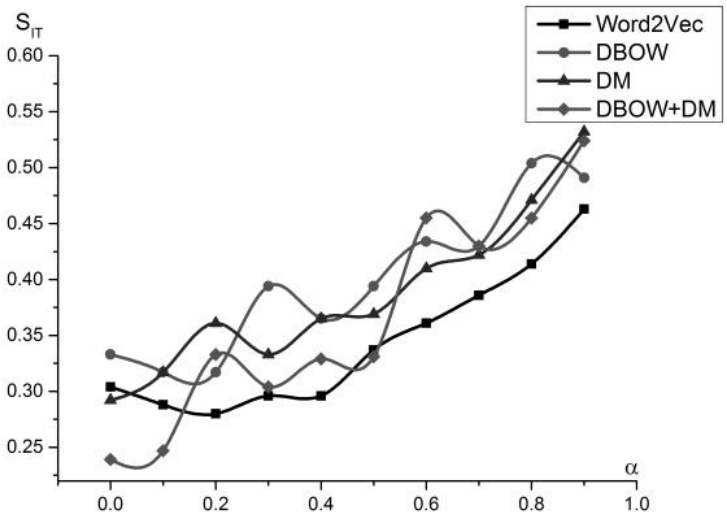

Рис. 4. Динаміка зміни оцінки задачі вставки відносно регулятивного параметра α для підходу PAV

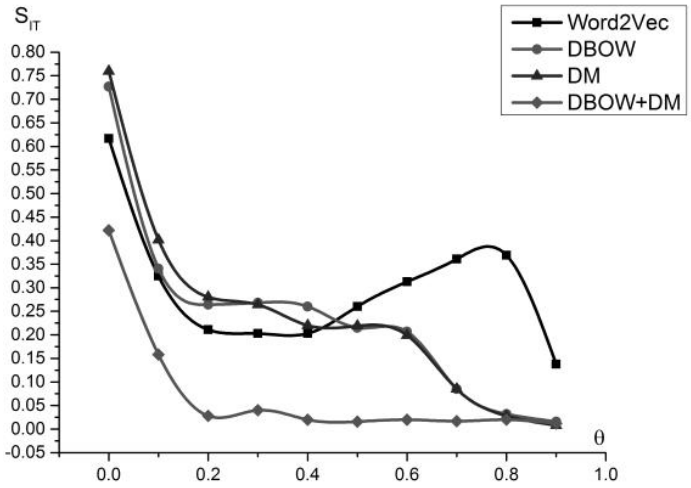

Рис. 5. Динаміка зміни оцінки задачі вставки відносно порогового значення θ для підходу MSV

Висновки

На основі аналізу отриманих результатів, можна зробити такі висновки:

— найбільш ефективною комбінацією моделі, підходу до побудови графа і його параметрів є граф семантичної схожості MSV з пороговим значенням $\theta = 0$, що використовує модель DM;

— граф семантичної схожості SSV показав найгірші результати для типових задач оцінки ефективності роботи методу розрахунку когерентності тексту, що вказує на наявність багаторазового зв'язку між реченнями тексту, написаного природною українською мовою;

— високі показники роботи графа PAV для значення параметра $\alpha > 0,5$ вказують на доцільність урахування спільних термів речень;

— динаміка спаду результатів зі збільшенням порогового значення θ для графа MSV вказує на необхідність урахування зв'язку між всіма реченнями тексту, незалежно від відстані між ними; відсутність поступового зменшення результатів для моделі Word2Vec можна пояснити фактором випадкового вибору вектору для слів, які відсутні у словнику навченої моделі;

— комбінація моделей DBOW+DM виявилась ефективною лише для графів PAV і SSV, що враховують одноразовий зв'язок між реченнями; такий результат можна спробувати трактувати як перенавчання комбінованої моделі, адже модель не здатна враховувати зв'язки між реченнями, що розташовані не поруч одне з одним;

— наявність багаторазового зв'язку та спільних термів між реченнями дозволяють зробити висновок про доцільність урахування *елементів міжфразової єдності* (синоніми, антоніми, гіпоніми, кореферентні зв'язки) для покращення результатів роботи методу ГСС, що і пропонується виконати в наступних дослідженнях.

1. Raymond E.S. The new hacker's dictionary. Mit Press, 1996. 568 p.

2. Publications — The Stanford Natural Language Processing Group. URL: https://nlp.stanford.edu/pubs (дата звернення: 08.12.2018).

3. Publications — Google AI. URL: https://ai.google/research/pubs (дата звернення: 08.12.2018).

4. Homepage: lang-uk. URL: http://lang.org.ua (дата звернення: 02.12.2018).

5. Лєднік О.С. Когезія та когерентність як категорії зв'язного тексту. *Науковий часопис Національного педагогічного університету імені М. П. Драгоманова. Серія 10: Проблеми граматики і лексикології української мови*. 2010. Вип. 6. С. 119–123.

6. Barzilay R., Lapata M. Modeling local coherence: An entity-based approach. *Computational Linguistics*. 2008. Vol. 34, No 1. P. 1–34.

7. Guinaudeau C., Strube M. Graph-based local coherence modeling. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. 2013. Vol. 1. P. 93–103.

8. Li J., Hovy E. A model of coherence based on distributed sentence representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014. P. 2039–2048.

9. Cui B., Li Y., Zhang Y., Zhang Z. Text Coherence Analysis Based on Deep Neural Network. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 2017. P. 2027–2030.

10. Putra J.W.G., Tokunaga T. Evaluating text coherence based on semantic similarity graph. *Proceedings of TextGraphs-11: the Workshop on Graph-based Methods for Natural Language Processing*. 2017. P. 76–85.

11. Le Q., Mikolov T. Distributed representations of sentences and documents. *International Conference on Machine Learning*. 2014. P. 1188–1196.

12. Погорілий С.Д., Крамов А.А. Автоматизована екстракція структурованої інформації з множини веб-сторінок. *Проблеми програмування.* 2018. № 2–3. С. 149–158.

13. Pogorilyy S., Kramov A. Automated extraction of structured information from a variety of web pages. *Proceedings of the 11th International Conference of Programming UkrPROG 2018*. Kyiv, 2018. P. 149–158.

14. Science Parse Server. URL: https://github.com/allenai/science-parse/blob/master/server/README.md (дата звернення: 08.12.2018).

15. gensim: Topic modelling for humans. URL: https://radimrehurek.com/gensim (дата звернення: 08.12.2018).

Надійшла до редакції 13.12.2018