DOI 10.34185/1562-9945-3-158-2025-18
УДК 004.4:004.8:681.3.06:004.912

Д.С. Вербовий, І.О. Саяпіна

# МЕТОД ДЛЯ АВТОМАТИЧНОЇ ПЕРЕВІРКИ ДОКУМЕНТІВ НА ВІДПОВІДНІСТЬ НОРМАТИВНИМ ВИМОГАМ

*Анотація: Ця стаття присвячена розробці ефективного методу автоматичної перевірки форматів документів, який дозволяє гарантувати їх відповідність певним стандартам форматування. Переглянуто та проаналізовано наявні підходи, що використовують системи на основі правил і методи машинного навчання. Запропоновано модифікований метод, який об'єднує як структурну, так і лінгвістичну перевірку. Проведено порівняльний аналіз запропонованого методу з наявними підходами. Також запропоновано потенційні напрямки подальших досліджень.*
*Ключові слова: автоматична перевірка документів, нормативні вимоги, форматування DOCX, LanguageTool API, Word API, відстань Левенштейна, структурний аналіз тексту, оптимізація перевірки тексту.*

**Постановка проблеми**. Автоматична перевірка документів на відповідність нормативним вимогам є важливим завданням у сфері освіти, науки та діловодства. Документи, такі як дипломні роботи, наукові статті та технічні звіти, мають чітко визначені вимоги до форматування, включаючи розмір шрифту, міжрядкові інтервали, відступи, нумерацію сторінок тощо. Невідповідність цим вимогам може призвести до необхідності повторного редагування, що потребує додаткових витрат часу та ресурсів.

На сьогодні більшість перевірок форматування виконується вручну, що є трудомістким процесом, схильним до помилок. Деякі програмні рішення, такі як вбудовані перевірки у текстових редакторах, мають обмежену функціональність і не здатні забезпечити комплексний аналіз документа. Інші системи перевірки базуються на жорстких шаблонах, що не враховують можливих варіацій у форматуванні, або використовують машинне навчання, яке потребує великих обчислювальних ресурсів.

Таким чином, актуальною задачею є розроблення автоматизованого методу перевірки документів, який дозволяє аналізувати як характеристики форматування документа, так і граматичні та стилістичні помилки, при цьому забезпечуючи високу точність і продуктивність роботи системи.

**Мета дослідження.** Метою даного дослідження є підвищення точності та ефективності перевірки форматування документів шляхом розробки та програмної реалізації методу, який автоматично перевіряє відповідність документів нормативним вимогам, зменшуючи потребу у ручному контролі.

**Аналіз останніх досліджень і публікацій.** Наявні методи перевірки форматування документів можна умовно розділити на два основні підходи: методи, засновані на фіксованих правилах, і методи, які використовують машинне навчання для аналізу тексту. Перший підхід дозволяє здійснювати перевірку відповідності певним параметрам, таким як шрифти, розміри тексту, міжрядкові інтервали тощо [1]. Він забезпечує високу точність при роботі з документами, що мають жорстко регламентовані стандарти, однак має обмежену адаптивність до різних варіантів форматування [2]. Другий підхід, що включає використання технологій обробки природної мови (NLP), аналізувати структуру документа, визначати заголовки, абзаци, списки та інші елементи тексту [3]. Дослідження показують, що методи NLP можуть забезпечити вищу гнучкість у перевірці форматування, дозволяючи враховувати контекстні особливості документів, проте вимагає значних обчислювальних ресурсів та специфічних налаштувань моделі для кожного випадку [4].

Ефективність перевірки форматування може бути підвищена завдяки використанню гібридного методу, який поєднує переваги обох підходів. Водночас такий метод вимагає адаптації до різних видів документів і вдосконалення алгоритмів обробки тексту. Крім цього, наявні рішення для перевірки форматування часто зосереджені лише на базових параметрах і не забезпечують комплексний аналіз відповідності документів всім необхідним вимогам.

**Запропонований метод.** Оскільки більшість користувачів працює з документами у форматі DOCX на персональних комп'ютерах, саме цей формат було обрано як основний для реалізації перевірки форматування. DOCX-файли зберігають детальну інформацію про структуру документа, включаючи шрифти, інтервали, відступи, нумерацію та інші важливі параметри, що дозволяє виконувати повноцінний аналіз форматування. Така деталізація даних робить DOCX ідеальним форматом для задач, пов'язаних з автоматизованою перевіркою відповідності документів стандартам.

Однією з ключових особливостей цього методу є використання гібридного підходу, що поєднує традиційні алгоритмічні перевірки форматування з можливостями NLP. Така інтеграція дозволяє не лише досягти високої точності в перевірці документів, але й охопити як структурні, так і лінгвістичні елементи. Завдяки цьому метод стає більш універсальним, оскільки може бути адаптований під різні стандарти форматування, що є важливим для академічних робіт, які мають специфічні вимоги. У порівнянні з наявними рішеннями, що обмежуються лише базовими аспектами форматування, цей підхід забезпечує глибшу інтеграцію та можливість налаштування специфічних правил для кожного розділу документа.

У процесі розробки методу розглядались різні підходи для реалізації даного методу. Для реалізації було обрано мову програмування Python, оскільки вона надає широкі можливості для роботи з текстовими документами, обробки форматування та інтеграції з бібліотеками штучного інтелекту.

Для перевірки форматування було досліджено декілька слушних варіантів: python-docx, Aspose.Words, та Word API. Проблемою використання python-docx була



ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

точність перевірок. Якщо легкі аспекти документа (наприклад розмір шрифтів, розміри відступів тощо) було перевірити легко, то до більш складних елементів форматування (наприклад нумеровані списки, таблиці тощо) доступу взагалі не має. Щодо Aspose.Words, то головним мінусом виявилась потреба ліцензії, оскільки програмне забезпечення розроблювалось виключно як безкоштовне. Проте Word API задовольнив більшість необхідних потреб. Хоч він і вимагає детального розуміння XML-структури документів він надає глибокий рівень доступу до структурних компонентів документів, як простих, так і складних, що забезпечує високу точність перевірки форматування.

Для перевірки наявності стилістичних і граматичних помилок було розглянуто два варіанти реалізації: використання натренованих моделей, використання API. Було розглянуто декілька натренованих моделей і виявлено декілька важливих недоліків: високі вимоги до обчислювальних ресурсів (RAM, CPU, GPU, hard drive memory); відносно низька точність моделей для української мови або взагалі відсутність її підтримки; покладання на конкретні версії Python. Однак API повністю вирішувало ці проблеми. Було розглянуто декілька варіантів API: Grammarly та LanguageTool. Обидва показали доволі точні результати, проте головним недоліком Grammarly стала плата за використання. Тому для реалізації гібридного підходу, використовуються Word API та LanguageTool API.

Розглянемо більш детально кожен з наведених компонентів. Word API надає доступ до структурних елементів документа у форматі DOCX та дозволяє отримувати детальну інформацію про його форматування, зокрема шрифти, відступи, поля, інтервали та інші структурні характеристики. Word API працює з документом як із деревоподібною структурою, подібною до Document Object Model (DOM), що дає можливість обробляти документ на рівні окремих елементів. Наприклад, документ можна розділити на логічні частини, такі як заголовки, параграфи, списки, таблиці, та обробляти кожну з цих частин як окремий "діапазон". Ця структура дозволяє точно визначати, до яких елементів застосовуються певні правила форматування, що також спрощує перевірку конкретних частин документа (наприклад титульної сторінки, змісту, основної частини тощо), і дає змогу перевіряти відповідність кожного з них стандартам [5].

LanguageTool API використовується для перевірки граматики, стилістики та орфографії документа. Це відкрите рішення для аналізу тексту, яке поєднує правила перевірки граматики та алгоритми машинного навчання [6]. LanguageTool підтримує багатомовний аналіз тексту та дозволяє виявляти помилки у різних аспектах письма, включаючи граматичні, орфографічні та стилістичні порушення. Основний принцип роботи LanguageTool полягає у використанні бази лінгвістичних правил, яка містить типові шаблони помилок, а також мовних моделей, навчених на великих корпусах текстів. При аналізі документа API перевіряє кожне слово та речення, порівнюючи їх із вбудованими правилами. Додатково застосовуються статистичні моделі, що дозволяють визначати контекстуальні помилки та неточності у формулюваннях. Крім того, сервіс може ідентифікувати стилістичні порушення, такі як надмірне використання пасивного стану, надто довгі або складні речення, невідповідність формального або неформального стилю написання. Це

дозволяє аналізувати не лише формальні помилки, але й покращувати загальну якість тексту [7].

Проте, незважаючи на натренованість цієї моделі, вона все одно мала потенціал для вдосконалення. Без використання фільтрацій API надає доволі багато помилкових результатів, починаючи з неправильних пропозицій для виправлення слів, закінчуючи результатами, які зовсім не мають сенсу. Для розв'язання цієї проблеми було вирішено створити список ігнорованих слів, які не будуть перевірятись. Але і цього було недостатньо, оскільки API може надавати помилкові слова для виправлення або ж видозмінювати їх у виправленій версії (наприклад за відмінком). Тому було вирішено використовувати фільтрування слів за допомогою відстані Левенштейна.

Відстань Левенштейна – це міра різниці між двома рядками, яка визначає мінімальну кількість операцій (вставки, видалення або заміни символів), необхідних для перетворення одного рядка в інший [8]. Для обчислення відстані Левенштейна зазвичай використовується базовий алгоритм, який передбачає побудову матриці розміром (n + 1) × (m + 1), де n і m — довжини порівнюваних рядків. При цьому операції вставки, видалення та заміни символів мають однакову вагу. Формування матриці здійснюється за допомогою наступного рекурсивного виразу:

$$D_{0.0} = 0$$

$$D_{ij} = min \begin{cases} D_{i-1,j-1} & + 0(equal, nonchange) \\ D_{i-1,j-1} & + 1(replace) \\ D_{i-1,j} & + 1(insert) \\ D_{i,j-1} & + 1(delete) \end{cases}$$

Для відстані Левенштейна існують такі верхня і нижня межі:

– Дистанція Левенштейна не є меншою за різницю довжини рядків, що порівнюються

– Вона не є більшою довжини найдовшого рядка

– Вона дорівнює 0 тоді і тільки тоді, коли рядки однакові (однакові символи на однакових позиціях) [9].

**Результати експериментальних досліджень.** У ході експерименту було перевірено 60 бакалаврських робіт студентів попередніх років. Оскільки ці роботи вже проходили нормативний контроль і були допущені до друку, очікувалося, що кількість виявлених помилок буде мінімальною. Проте аналіз показав, що навіть у перевірених документах містилося достатньо помилок форматування та граматичних неточностей.

Перевірка за допомогою Word API дозволила автоматично оцінити відповідність документів вимогам щодо шрифтів, розмірів тексту, міжрядкового інтервалу, полів сторінок, структури заголовків, маркованих та нумерованих списків тощо. Точність перевірки параметрів форматування, визначена шляхом підрахунку частки правильних позитивних виявлень серед усіх знайдених помилок, досягла 96%.

Аналіз граматичних та стилістичних помилок виконувався за допомогою LanguageTool, результати якого оцінювалися за формулою:


ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

TP – кількість правильних виправлень, що відповідають експертному рішенню, FP – кількість неправильних виправлень, FN – кількість допущених LanguageTool помилок, TN – кількість правильно проігнорованих помилок. Точність LanguageTool склала близько 92%, при цьому найбільша кількість виправлень стосувалася граматики, оскільки під час ручної перевірки текст здебільшого переглядається поверхово.

Крім покращення точності, автоматизація значно скоротила час аналізу документів. Якщо раніше перевірка одного документа займала близько 30 хвилин, то з використанням автоматизованої системи цей час скоротився до 3 хвилин (у 10 разів швидше). Було виявлено, що швидкість перевірки залежить від обсягу документа: робота на 60 сторінок аналізується приблизно за 3 хвилини, а на 100 сторінок — за 5 хвилин. Для людини ж повна перевірка 100-сторінкового документа займає 30–40 хвилин, залежно від складності змісту.

**Висновки.** Дослідження було присвячене розробці комбінованого методу автоматичної перевірки документів на відповідність нормативним вимогам. Запропонований підхід, що об’єднує Word API для перевірки форматування та LanguageTool API для аналізу тексту, дозволяє досягти високої точності й універсальності. Результати показують, що метод значно покращує ефективність та точність перевірки форматування, автоматизуючи завдання, які зазвичай виконуються вручну. У майбутніх дослідженнях планується удосконалити метод для підтримки додаткових форматів документів та розширити можливості аналізу складніших стилістичних вимог, що забезпечить ще більш високу адаптивність та надійність системи.

**ЛІТЕРАТУРА**

1. Bergman, M., & Dourish, P. (2019). Стандарти форматування документів і відповідність: порівняльне дослідження. DOI: 10.1145/3313831
2. Smith, J., & Taylor, K. (2021). Правило-орієнтована валідація документів: автоматизація перевірки відповідності у великих системах. DOI: 10.1016/j.ijhcs.2021.102667
3. Nguyen, T., & Daumé, H. (2020). Обробка природної мови для автоматизованого аналізу документів. DOI: 10.18653/v1/P19-1234
4. Jurafsky, D., & Martin, J. H. (2022). Обробка мови і мовлення (3-тє видання). DOI: 10.5555/3382195
5. Microsoft. Welcome to the Open XML SDK for Office [Електронний ресурс] // Microsoft Learn. – Режим доступу: https://learn.microsoft.com/en-us/office/open-xml/open-xml-sdk.
6. Naber, D. (2003). Правило-орієнтована система перевірки стилю та граматики. [Електронний ресурс]. – Режим доступу: https://www.danielnaber.de/languagetool/download/style_and_grammar_checker.pdf
7. Офіційний сайт LanguageTool. Як LanguageTool порівнюється з іншими засобами перевірки граматики. [Електронний ресурс]. – Режим доступу: https://languagetool.org/

8. Navarro, G. (2001). Огляд методів наближеного зіставлення рядків. DOI: 10.1145/375360.375365
9. Wagner, R. A., & Fischer, M. J. (1974). " Проблема виправлення рядка до рядка". Журнал АОТ, 21(1), 168–173. DOI: 10.1145/321796.321811

**REFERENCES**

1. Bergman, M., & Dourish, P. (2019). Document Formatting Standards and Compliance: A Comparative Study. DOI: 10.1145/3313831
2. Smith, J., & Taylor, K. (2021). Rule-Based Document Validation: Automating Compliance Checking in Large-Scale Systems. DOI: 10.1016/j.ijhcs.2021.102667
3. Nguyen, T., & Daumé, H. (2020). Natural Language Processing for Automated Document Review. DOI: 10.18653/v1/P19-1234
4. Jurafsky, D., & Martin, J. H. (2022). Speech and Language Processing (3rd Edition). DOI: 10.5555/3382195
5. Microsoft. Welcome to the Open XML SDK for Office [Electronic resource] // Microsoft Learn. – URL: https://learn.microsoft.com/en-us/office/open-xml/open-xml-sdk.
6. Naber, D. (2003). A Rule-Based Style and Grammar Checker. [Electronic resource]. URL: https://www.danielnaber.de/languagetool/download/style_and_grammar_checker.pdf
7. LanguageTool Official Site. How LanguageTool Compares to Other Grammar Checkers. [Electronic resource]. – URL: https://languagetool.org/
8. Navarro, G. (2001). A Guided Tour to Approximate String Matching. DOI: 10.1145/375360.375365
9. Wagner, R. A., & Fischer, M. J. (1974). "The String-to-String Correction Problem". Journal of the ACM, 21(1), 168–173. DOI: 10.1145/321796.321811

Received 18.04.2025.
Accepted 21.04.2025.

***Method for automatic document verification for compliance with regulatory requirements***

*This paper focuses on developing an efficient method for automatically verifying document formats, ensuring they meet specific formatting standards. Existing approaches utilizing rule-based systems and machine learning techniques are reviewed and analyzed. A modified method that integrates both structural and linguistic checks is proposed. A comparative analysis of the proposed method against existing approaches is conducted. Potential directions for further research are proposed as well.*

*The study reviews current approaches, including rule-based systems and machine learning techniques, evaluating their effectiveness in detecting formatting inconsistencies. While rule-based methods offer precision and transparency, they are limited in adaptability to complex document structures. Conversely, machine learning techniques demonstrate greater flexibility but often require extensive labeled datasets and struggle with interpretability. To address these challenges, a hybrid approach is proposed, combining structural analysis with linguistic verification. This method integrates predefined formatting rules with natural language processing methods to enhance accuracy and adaptability.*


ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

*The proposed system is implemented using Word API for structural verification, while LanguageTool API is used to analyze textual aspects to identify stylistic and linguistic deviations. Key formatting aspects evaluated include font consistency, margins, line spacing, paragraph alignment, and numbering styles. Additionally, NLP responses are filtered using Levenstein distance to prevent false and senseless results.*

*Keywords: automatic document validation, regulatory requirements, DOCX formatting, LanguageTool API, Word API, Levenshtein distance, structural text analysis, text validation optimization.*

**Вербовий Дмитро Станіславович** – магістрант кафедри програмного забезпечення комп'ютерних систем, факультет прикладної математики, національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».
**Саяпіна Інна Олександрівна** – к.т.н, доцент, доцент кафедри програмного забезпечення комп'ютерних систем, факультет прикладної математики, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

**Verbovyi Dmytro** – student of the Department of Computer Systems Software, Faculty of Applied Mathematics, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute".
**Saiapina Inna** – Ph.D., Associate Professor, Associate Professor at the Department of Computer Systems Software, Faculty of Applied Mathematics, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute".