DEVELOPMENT OF AN AUTOMATIC TEXT ANALYSIS APPLICATION USING C#

Gîlcă Gheorghe, Lecturer Phd., *"Constantin Brâncuși" University from Târgu Jiu, ROMANIA*

ABSTRACT: The paper presents the development of a text analyzer implemented in C#, designed for the automated processing of Romanian-language texts. The system provides key statistics such as the total number of characters, words, sentences, the average word length, and word frequency. The tool is built using Windows Forms technology and integrates regular expressions for linguistic processing, following methodologies established in the relevant literature [1], [2]. The system architecture, the algorithms employed, experimental results, and future development directions are also presented.

KEY WORDS: text analysis, C#, WinForms, textual statistics, Regex, word frequency

1. INTRODUCTION

Automatic text analysis is a fundamental component of natural language processing (NLP), with applications in research, education, the legal domain, journalism, and digital marketing. The massive growth of textual content requires tools capable of extracting relevant information quickly and efficiently. According to Jurafsky & Martin [1], statistical analysis serves as a preliminary stage for advanced NLP models.

In the absence of customizable desktop tools, many existing applications rely on online platforms, where text confidentiality becomes problematic. A local tool provides flexibility, control, and opportunities for future expansion, as recommended in the works of Manning & Schütze [2].

2. LITERATURE REVIEW

Modern NLP systems—such as NLTK and spaCy—are widely used in both academic and industrial environments [3][4]. These tools provide tokenization, part-of-speech tagging, semantic analysis, and word vector generation. However, they can be difficult for the general public to use.

Web-based tools offer limited statistics and raise security concerns. Studies in corpus linguistics [5] and text classification [6] highlight the importance of local data preprocessing, which makes a desktop analyzer extremely valuable.

The task of teaching and learning algorithms represents one of the main challenges in the computer science education community, mainly due to the complexity of intuitively understanding how they work [7].

2.1. Evolution and Design of C#

Visual C# is the primary object-oriented programming language developed by Microsoft for the .NET platform, introduced in 2000 as part of the .NET Framework initiative. C# was designed to combine the simplicity of high-level languages with the power and control of modern programming languages, offering advanced support for safe coding practices, automatic memory management, and static typing.

The official documentation describes C# as a "modern, simple, powerful, and productive" language, intended for the development of Windows, web, cloud, and mobile applications [8].

The language is part of a broader ecosystem integrated into the Visual Studio development environment. Visual Studio provides a compiler, debugging tools, a visual designer, and support for multi-layer projects, making C# a preferred choice for developers building enterprise applications. According to Microsoft, the .NET platform and Visual C# were designed to promote interoperability, scalability, and the reuse of software components [8], [9].

2.2. Core Features and Advantages of Visual C#

Visual C# is based on the object-oriented programming (OOP) paradigm, adopting concepts such as inheritance, encapsulation, polymorphism, and abstraction—fundamental principles shared by major languages like Java and C++ [10]. Additionally, C# introduces modern extensions such as events, delegates, lambda expressions, LINQ (Language Integrated Query), generics, and async/await programming, all of which facilitate the development of more expressive and robust code [8]. Another major advantage of the C# language is its integration with the Common Language Runtime (CLR), which enables code execution in a controlled environment, independent of the underlying physical platform, providing automatic memory management and protection against illegal memory access errors [9]. This approach makes it safer and more efficient compared to older languages that lack a managed runtime.

Regarding graphical user interface development, Visual C# offers native support for Windows Forms, WPF, and ASP.NET, being widely used in educational and industrial environments for rapid prototyping and desktop application development [8], [11].

The evolution of the language continues steadily, with each new version introducing additional functionalities such as pattern matching, nullable reference types, and record types, in line with community recommendations and Microsoft's strategic direction [8].

2.3. Educational Aspects and Learning Visual C#

Teaching and learning Visual C# has been thoroughly examined within educational research. Recognized for its clarity and intuitive structure, C# is frequently selected for both introductory programming courses and for teaching advanced topics, including object-oriented principles, data structures, and software design patterns.

Regarding accessibility, several studies indicate that C# is well-suited for beginners due to its clean, readable syntax and the availability of robust standard libraries [12].

An analysis in the paper [13] attempts to investigate the ways in which the use of open-source AI tools can modify and influence the educational activities of students in technical universities.

In contrast with languages that feature more complex syntactic rules, such as C++, C# provides an effective balance between simplicity and professional-level functionality, making it a strong and adaptable choice for academic instruction.

3. CREATING AN AUTOMATIC TEXT ANALYSIS APPLICATION USING C#

By using the .NET platform and the C# language, the application enables fast and efficient text processing, providing users with relevant statistics such as the total number of words, sentences, term frequency, and lexical distribution.The implementation of the graphical interface in Windows Forms facilitates access for users without advanced technical knowledge, allowing text analysis through an intuitive interaction. .NET regular expressions and LINQ components are used for tokenization and aggregation, ensuring accurate and high-performance processing.

Overall, the development of such an application demonstrates the versatility of the

C# language and its relevance in projects focused on the automatic analysis of natural language.

The source code of the created application is presented below. It contains the following functions regarding text analysis: counting the number of words, characters, sentences, lines, average word length, average words per sentence, but also the 20 most common words.

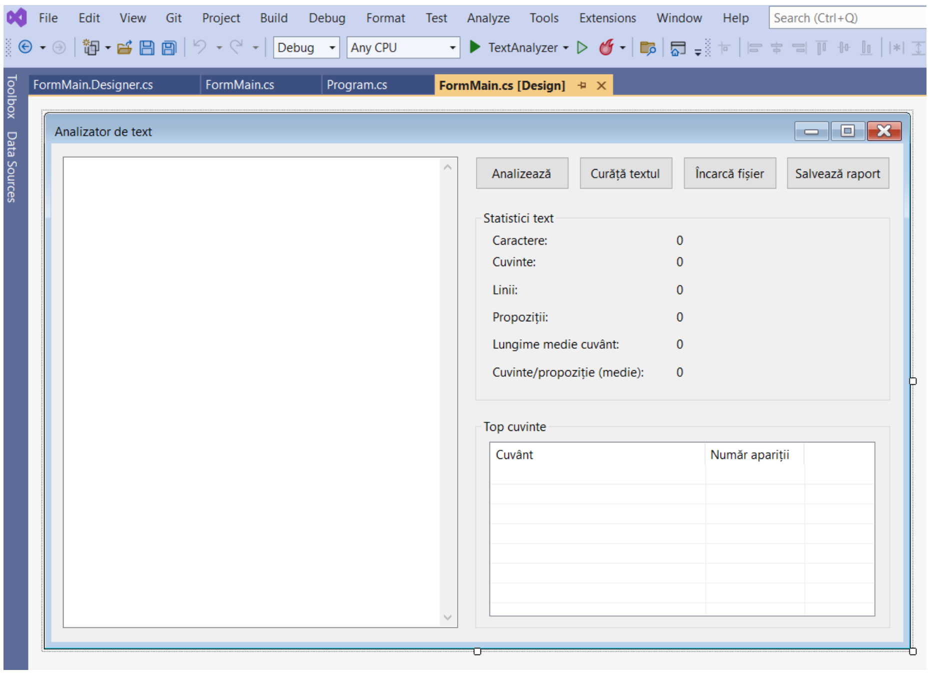
```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text.RegularExpressions;
using System.Windows.Forms;

namespace TextAnalyzer
{
    public partial class FormMain : Form
    {
        public FormMain()
        {
            InitializeComponent();
        }

        private void btnAnalyze_Click(object sender, EventArgs e)
        {
            string text = txtInput.Text;

            if (string.IsNullOrWhiteSpace(text))
            {
                MessageBox.Show("Introduceți sau încărcați un text pentru analiză.",
                    "Atenție", MessageBoxButtons.OK, MessageBoxIcon.Information);
                return;
            }

            AnalyzeText(text);
        }

        private void AnalyzeText(string text)
        {
            // Caractere
            int charCount = text.Length;

            // Linii (numărăm doar liniile non-goale)
            var lines = text
                .Split(new[] { "\r\n", "\n" }, StringSplitOptions.None)
                .Where(l => !string.IsNullOrWhiteSpace(l))
                .ToList();
            int lineCount = lines.Count;

            // Cuvinte (folosim regex pe litere)
            var wordMatches = Regex.Matches(text.ToLower(), @"\p{L}+");
            List<string> words = wordMatches
                .Cast<Match>()
                .Select(m => m.Value)
                .ToList();
            int wordCount = words.Count;

            // Propoziții (split după . ! ?)
            var sentenceParts = Regex.Split(text, @"[\.!\?]+")
                .Select(s => s.Trim())
                .Where(s => !string.IsNullOrWhiteSpace(s))
                .ToList();
            int sentenceCount = sentenceParts.Count;

            // Media lungimii cuvintelor
            double avgWordLen = wordCount > 0
                ? words.Average(w => w.Length)
                : 0.0;

            // Media cuvintelor pe propoziție
            double avgWordsPerSentence = sentenceCount > 0
                ? (double)wordCount / sentenceCount
                : 0.0;

            // Actualizăm etichetele
            lblCharsValue.Text = charCount.ToString();
            lblWordsValue.Text = wordCount.ToString();
            lblLinesValue.Text = lineCount.ToString();
            lblSentencesValue.Text = sentenceCount.ToString();
            lblAvgWordLenValue.Text = avgWordLen.ToString("0.00");
            lblAvgWordsPerSentenceValue.Text = avgWordsPerSentence.ToString("0.00");

            // Top cuvinte
            UpdateTopWords(words);
        }

        private void UpdateTopWords(List<string> words)
        {
            lvTopWords.Items.Clear();

            if (words.Count == 0)
                return;

            var top = words
                .GroupBy(w => w)
                .Select(g => new { Word = g.Key, Count = g.Count() })
                .OrderByDescending(x => x.Count)
                .ThenBy(x => x.Word)
                .Take(20)
                .ToList();

            foreach (var item in top)
            {
                var li = new ListViewItem(item.Word);
                li.SubItems.Add(item.Count.ToString());
                lvTopWords.Items.Add(li);
            }
        }

        private void btnClear_Click(object sender, EventArgs e)
        {
            txtInput.Clear();
            lvTopWords.Items.Clear();

            lblCharsValue.Text = "0";
            lblWordsValue.Text = "0";
            lblLinesValue.Text = "0";
            lblSentencesValue.Text = "0";
            lblAvgWordLenValue.Text = "0";
            lblAvgWordsPerSentenceValue.Text = "0";
        }

        private void btnLoad_Click(object sender, EventArgs e)
        {
            using (var ofd = new OpenFileDialog())
            {
                ofd.Filter = "Text files (*.txt)|*.txt|All files (*.*)|*.*";
                ofd.Title = "Deschide fișier text";

                if (ofd.ShowDialog() == DialogResult.OK)
                {
                    try
```

```
          {
            string text = File.ReadAllText(ofd.FileName);
            txtInput.Text = text;
          }
          catch (Exception ex)
          {
            MessageBox.Show("Eroare la citirea fișierului:\r\n" + ex.Message,
              "Eroare", MessageBoxButtons.OK, MessageBoxIcon.Error);
          }
        }
      }
    }

    private void btnSaveReport_Click(object sender, EventArgs e)
    {
      // Generează un mic raport text cu rezultatele
      string report = GenerateReport();

      using (var sfd = new SaveFileDialog())
      {
        sfd.Filter = "Text files (*.txt)|*.txt|All files (*.*)|*.*";
        sfd.Title = "Salvează raportul";
        sfd.FileName = "RaportAnalizaText.txt";

        if (sfd.ShowDialog() == DialogResult.OK)
        {
          try
          {
            File.WriteAllText(sfd.FileName, report);
            MessageBox.Show("Raport salvat cu succes.", "Info",
              MessageBoxButtons.OK, MessageBoxIcon.Information);
          }
          catch (Exception ex)
          {
            MessageBox.Show("Eroare la salvarea raportului:\r\n" + ex.Message,
              "Eroare", MessageBoxButtons.OK, MessageBoxIcon.Error);
          }
        }
      }
    }

    private string GenerateReport()
    {
      return
        "Raport analiză text\r\n" +
        "===================\r\n\r\n" +
        $"Caractere: {lblCharsValue.Text}\r\n" +
        $"Cuvinte: {lblWordsValue.Text}\r\n" +
        $"Linii: {lblLinesValue.Text}\r\n" +
        $"Propoziții: {lblSentencesValue.Text}\r\n" +
        $"Lungime medie cuvânt: {lblAvgWordLenValue.Text}\r\n" +
        $"Cuvinte/propoziție (medie): {lblAvgWordsPerSentenceValue.Text}\r\n\r\n" +
        "Top cuvinte:\r\n" +
        string.Join("\r\n", lvTopWords.Items
          .Cast<ListViewItem>()
          .Select(i => $"{i.Text}: {i.SubItems[1].Text}"));
    }
  }
}
```

Figure 1 shows the interface created in Visual Studio, containing 4 buttons, a window where the text to be analyzed is entered, a window where details about the analyzed text will appear, and several labels with details about the number of words, sentences, characters, etc.

Figure 2 shows the result generated by the application for the given text to be analyzed. The application provides information regarding: the number of characters in the text, the number of words, the number of lines (they must have the . sign at the end), the average length of characters/word, the average length of words / sentence.

Figure 3 shows how the application provides us with a report that we can save in a text file with all the related details (number of words, number of lines, number of characters, frequency of occurrence of certain words, etc.).

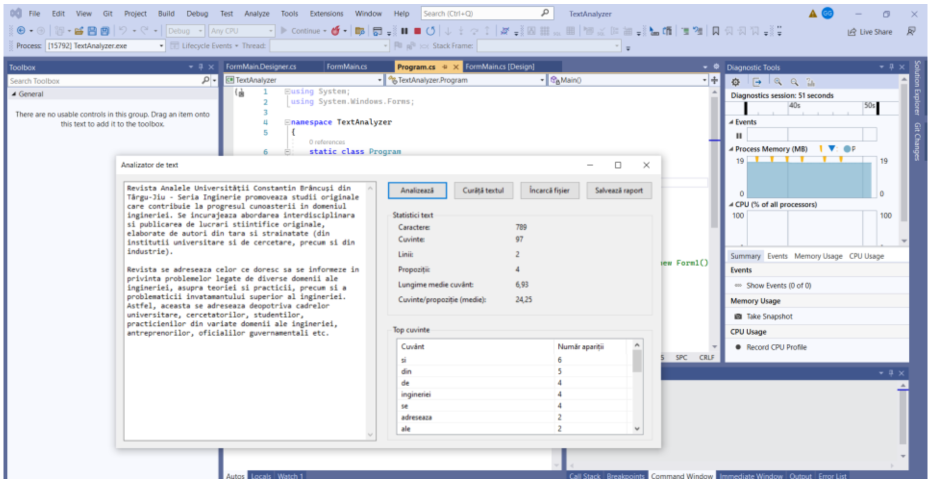


Statistici text

Caractere:	0
Cuvinte:	0
Linii:	0
Propoziții:	0
Lungime medie cuvânt:	0
Cuvinte/propoziție (medie):	0

Cuvânt	Număr apariții

Figure 1. Design form of the application



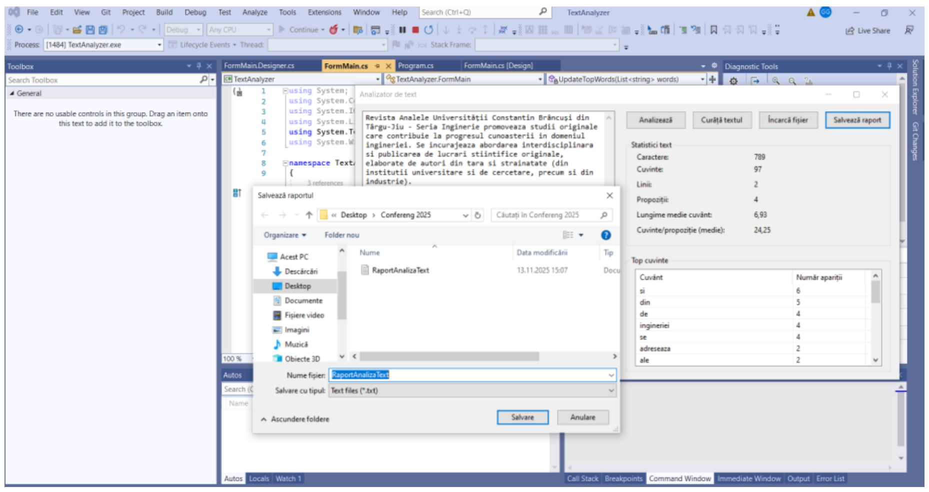
```
Revista Analele Universității Constantin Brâncuși din
Târgu-Jiu - Seria Inginerie promoveaza studii originale
care contribuie la progresul cunoasterii in domeniul
ingineriei. Se incurajeaza abordarea interdisciplinara
si publicarea de lucrari stiintifice originale,
elaborate de autori din tara si strainatate (din
institutii universitare si de cercetare, precum si din
industrie).

Revista se adreseaza celor ce doresc sa se informeze in
privinta problemelor legate de diverse domenii ale
ingineriei, asupra teoriei si practicii, precum si a
problematicii invatamantului superior al ingineriei.
Astfel, aceasta se adreseaza deopotriva cadrelor
universitare, cercetatorilor, studentilor,
practicienilor din variate domenii ale ingineriei,
antreprenorilor, oficialilor guvernamentali etc.
```

Statistici text

Caractere:	789
Cuvinte:	97
Linii:	2
Propoziții:	4
Lungime medie cuvânt:	6,93
Cuvinte/propoziție (medie):	24,25

Cuvânt	Număr apariții
si	6
din	5
de	4
ingineriei	4
se	4
adreseaza	2
ale	2

Figure 2. Results generated by the application after analyzing the given text



Nume fișier: RaportAnalizaText
Salvare cu tipul: Text files (*.txt)

Figure 3. Saving the generated report

4. CONCLUSIONS

The paper demonstrates that using C# and WinForms, an efficient, easy-to-use, fast and extensible text analysis tool can be developed. The system represents a solid starting point for advanced natural language processing applications.

The application can be used in:

• academia

• writing and proofreading

• journalism

• training for advanced NLP analysis.

REFERENCES

[1]D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/

[2]C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press, 1999.

[3]S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. O’Reilly Media, 2009.

[4]spaCy Documentation, “Industrial-Strength Natural Language Processing in Python.” [Online]. Available: https://spacy.io

[5]T. McEnery and A. Hardie, *Corpus Linguistics: Method, Theory and Practice*. Cambridge: Cambridge University Press, 2012.

[6]F. Sebastiani, “Machine Learning in Automated Text Categorization,” *ACM Computing Surveys*, vol. 34, no. 1, pp. 1–47, 2002.

[7]Adrian Runceanu, Mihaela-Ana Runceanu, “Challenges in teaching programming and algorithms”, INTED2016 Proceedings, 10th International Technology, Education and Development Conference Valencia, Spain. 7-9 March, 2016, ISBN: 978-84-608-5617-7 / ISSN: 2340-1079, Pages: 4120-4126 DOI: 10.21125/inted.2016.2003, WOS:000402738404019

[8]Microsoft Docs, “C# Programming Guide,” 2024. [Online]. Available: https://learn.microsoft.com/dotnet/csharp/

[9]Microsoft Docs, “.NET Architecture,” 2024. [Online]. Available: https://learn.microsoft.com/dotnet/standard/

[10] B. Eckel, *Thinking in C#*, Prentice Hall, 2003.

[11] Microsoft Docs, “Windows Forms Overview,” 2024. [Online]. Available: https://learn.microsoft.com/dotnet/desktop/winforms/

[12] Kölling, M., “Teaching introductory programming: a new approach using BlueJ and C#,” Journal of Computing Sciences in Colleges, 2015.

[13] Adrian Runceanu, “The role of artificial intelligence tools for teaching in technical universities”, EDULEARN25 Proceedings, pp. 6900-6905, ISBN: 978-84-09-74218-9, ISSN: 2340-1117, doi: 10.21125/edulearn.2025.1690, Conference name: 17th International Conference on Education and New Learning Technologies, 30 June-2 July, 2025, Location: Palma, Spain