

Звіт з аналізу тексту за допомогою Azure AI Language

Вступ

Даний звіт описує веб-додаток, розроблений для аналізу тексту з використанням Azure AI Language Service. Програма дозволяє виконувати різноманітні операції обробки природної мови (NLP), такі як розпізнавання іменованих сутностей, аналіз тексту та спеціалізований освітній аналіз. Dodatok побудований на ASP.NET Core з використанням MVC-архітектури та інтегрований із хмарними сервісами Microsoft Azure.

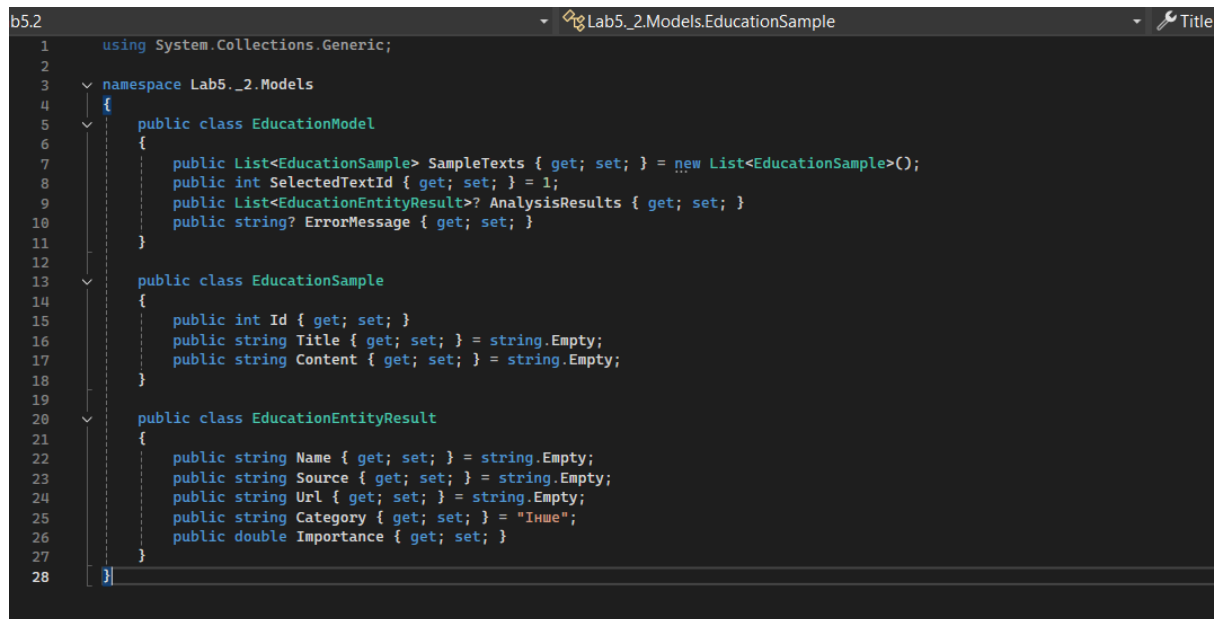
Основні функції програми

Програма надає можливості для аналізу іменованих сутностей (Named Entity Recognition), виявлення категорій (наприклад, люди, організації, локації, дати), підкатегорій (наприклад, політик, науковець), а також оцінки впевненості моделі. Також вона дозволяє виконувати загальний аналіз тексту з виявленням ключових сутностей, витягненням джерел та посилань, а також визначенням важливості кожної сутності. Додатково реалізований освітній аналіз, що включає спеціалізовану обробку навчальних текстів, класифікацію за категоріями (історія, математика, література) та демонстраційні приклади для тестування.

```
ab5.2 Lab5_2.Models.NamedEntityResult
1  using System.Collections.Generic;
2  using System.ComponentModel.DataAnnotations;
3
4  namespace Lab5_2.Models
5  {
6      public class NamedEntityModel
7      {
8          [Required(ErrorMessage = "Please enter text for analysis")]
9          public string InputText { get; set; } = string.Empty;
10         public List<NamedEntityResult>? Entities { get; set; }
11         public string? ErrorMessage { get; set; }
12     }
13
14     public class NamedEntityResult
15     {
16         public string Text { get; set; } = string.Empty;
17         public string Category { get; set; } = string.Empty;
18         public string SubCategory { get; set; } = string.Empty;
19         public double Confidence { get; set; }
20     }
21 }
```

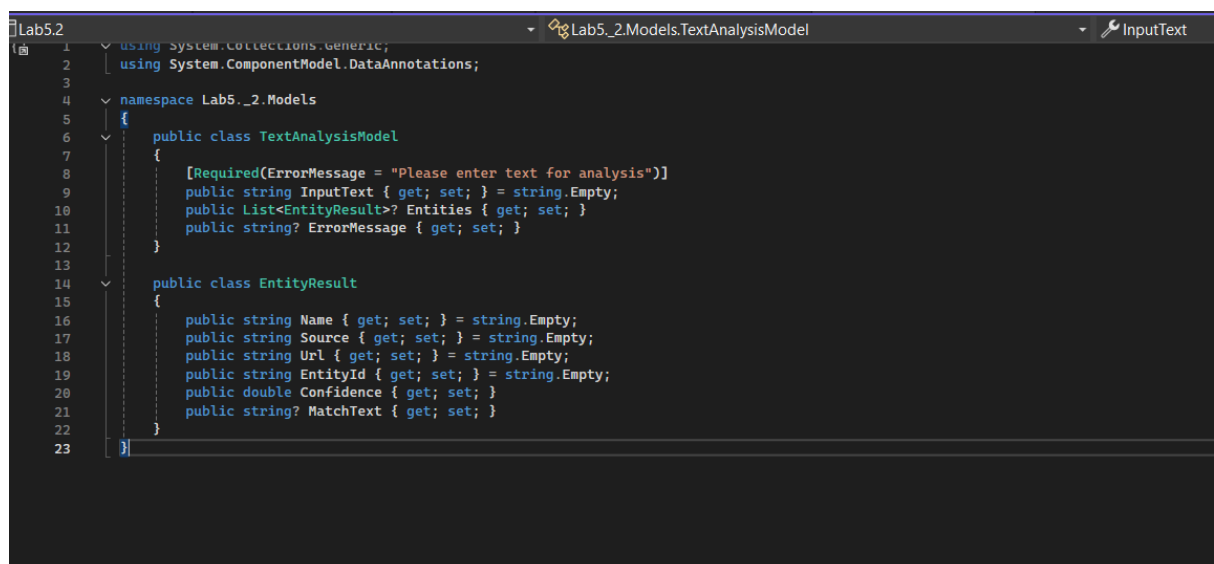
Технічна реалізація

Архітектура додатка базується на ASP.NET Core MVC з використанням Razor Pages для інтерфейсу, DI (Dependency Injection) для керування залежностями та Azure SDK для інтеграції з Text Analytics API. Основні компоненти включають моделі NamedEntityModel, TextAnalysisModel, EducationModel, ErrorViewModel, які відповідають за обробку різних типів аналізу. Основні сторінки інтерфейсу – Education.cshtml, NamedEntities.cshtml, Error.cshtml. Налаштування Azure Text Analytics реалізується через TextAnalyticsClient з використанням параметрів у appsettings.json.



```
1 using System.Collections.Generic;
2
3 namespace Lab5_2.Models
4 {
5     public class EducationModel
6     {
7         public List<EducationSample> SampleTexts { get; set; } = new List<EducationSample>();
8         public int SelectedTextId { get; set; } = 1;
9         public List<EducationEntityResult>? AnalysisResults { get; set; }
10        public string? ErrorMessage { get; set; }
11    }
12
13    public class EducationSample
14    {
15        public int Id { get; set; }
16        public string Title { get; set; } = string.Empty;
17        public string Content { get; set; } = string.Empty;
18    }
19
20    public class EducationEntityResult
21    {
22        public string Name { get; set; } = string.Empty;
23        public string Source { get; set; } = string.Empty;
24        public string Url { get; set; } = string.Empty;
25        public string Category { get; set; } = "Інше";
26        public double Importance { get; set; }
27    }
28 }
```

Рис. 2 реалізація Education Model



```
1 using System.Collections.Generic;
2 using System.ComponentModel.DataAnnotations;
3
4 namespace Lab5_2.Models
5 {
6     public class TextAnalysisModel
7     {
8         [Required(ErrorMessage = "Please enter text for analysis")]
9         public string InputText { get; set; } = string.Empty;
10        public List<EntityResult>? Entities { get; set; }
11        public string? ErrorMessage { get; set; }
12    }
13
14    public class EntityResult
15    {
16        public string Name { get; set; } = string.Empty;
17        public string Source { get; set; } = string.Empty;
18        public string Url { get; set; } = string.Empty;
19        public string EntityId { get; set; } = string.Empty;
20        public double Confidence { get; set; }
21        public string? MatchText { get; set; }
22    }
23 }
```

Рис. 2 реалізація Text Analysis Model

```

namespace Lab5._2.Controllers
{
    public class HomeController : Controller
    {
        private readonly TextAnalyticsClient _textAnalyticsClient;
        private readonly ILogger<HomeController> _logger;

        public HomeController(TextAnalyticsClient textAnalyticsClient, ILogger<HomeController> logger)
        {
            _textAnalyticsClient = textAnalyticsClient;
            _logger = logger;
        }

        public IActionResult Index()
        {
            return View(new TextAnalysisModel());
        }

        public IActionResult Education()
        {
            return View(new EducationModel
            {
                SampleTexts = GetSampleTexts()
            });
        }

        public IActionResult NamedEntities()
        {
            return View(new NamedEntityModel());
        }

        [HttpPost]
        public async Task<IActionResult> Analyze(TextAnalysisModel model)
        {
            if (!ModelState.IsValid) return View("Index", model);
        }
    }
}

```

```

[HttpPost]
public async Task<IActionResult> Analyze(TextAnalysisModel model)
{
    if (!ModelState.IsValid) return View("Index", model);

    try
    {
        Response<LinkedEntityCollection> response = await _textAnalyticsClient.RecognizeLinkedEntitiesAsync(model.InputText);
        model.Entities = response.Value
            .Select(e => {
                var match = e.Matches.FirstOrDefault();
                return new EntityResult
                {
                    Name = e.Name ?? "Без назви",
                    Source = e.DataSource ?? "Невідоме джерело",
                    Url = e.Url?.AbsoluteUri ?? string.Empty,
                    EntityId = e.DataSourceEntityId ?? string.Empty,
                    Confidence = match.ConfidenceScore,
                    MatchText = match.Text
                };
            })
            .ToList();
    }
    catch (Exception ex)
    {
        model.ErrorMessage = $"Помилка: {ex.Message}";
        _logger.LogError(ex, "Text analysis error");
    }

    return View("Index", model);
}

```

```

[HttpPost]
public async Task<IActionResult> AnalyzeEducation(EducationModel model)
{
    model.SampleTexts = GetSampleTexts();

    var selectedSample = model.SampleTexts.FirstOrDefault(s => s.Id == model.SelectedTextId);
    if (selectedSample == null)
    {
        model.ErrorMessage = "Будь ласка, виберіть текст для аналізу";
        return View("Education", model);
    }

    try
    {
        Response<LinkedEntityCollection> response = await _textAnalyticsClient.RecognizeLinkedEntitiesAsync(selectedSample.Content);
        model.AnalysisResults = response.Value
            .Select(e => {
                var match = e.Matches.FirstOrDefault();
                return new EducationEntityResult
                {
                    Name = e.Name ?? "Без назви",
                    Source = e.DataSource ?? "Невідоме джерело",
                    Url = e.Url?.AbsoluteUri ?? string.Empty,
                    Category = GetCategory(e.DataSource),
                    Importance = match.ConfidenceScore
                };
            })
            .ToList();
    }
}

```

```
        }
        catch (Exception ex)
        {
            model.ErrorMessage = $"Помилка аналізу: {ex.Message}";
            _logger.LogError(ex, "Education analysis error");
        }
    }

    return View("Education", model);
}

[HttpPost]
public async Task<IActionResult> AnalyzeNamedEntities(NamedEntityModel model)
{
    if (!ModelState.IsValid) return View("NamedEntities", model);

    try
    {
        Response<CategorizedEntityCollection> response = await _textAnalyticsClient.RecognizeEntitiesAsync(model.InputText);
        model.Entities = response.Value
            .Select(e => new NamedEntityResult
            {
                Text = e.Text,
                Category = e.Category.ToString(),
                SubCategory = e.SubCategory ?? "Немає",
                Confidence = e.ConfidenceScore
            })
            .ToList();
    }
    catch (Exception ex)
    {
        model.ErrorMessage = $"Помилка: {ex.Message}";
    }
}
```

```
private List<EducationSample> GetSampleTexts()
{
    return new List<EducationSample>
    {
        new EducationSample { Id = 1, Title = "Історичний текст", Content = "Kyiv was founded in the 5th century. In 882, Prince Oleg captured the city and made it the capital of Kievan Rus'." },
        new EducationSample { Id = 2, Title = "Науковий текст", Content = "Albert Einstein published the special theory of relativity in 1905. This theory changed the way we understand space and time." },
        new EducationSample { Id = 3, Title = "Географічний текст", Content = "The Dnieper is the longest river in Ukraine. It originates in Russia, flows through Belarus, and empties into the Black Sea." }
    };
}

private static string GetCategory(string? dataSource)
{
    return dataSource switch
    {
        "Wikipedia" => "Загальні знання",
        "Bing" => "Повукова інформація",
        _ => "Інше"
    };
}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
public IActionResult Error()
{
    return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
}
```

Рис. 3-7 реалізація Home Controller

Приклад роботи програми

Вхідні дані:
"Microsoft була заснована Біллом Гейтсом у 1975 році. Штаб-квартира компанії знаходиться в Редмонді, США."

Результати аналізу:

Сутність	Категорія	Підкатегорія	Впевненість
Microsoft	Організація	Технологічна компанія	0.98
Білл Гейтс	Персона	Закладник	0.95
1975 рік	Дата	-	0.90
Редмонд	Локація	Місто	0.93
США	Локація	Країна	0.99

Educational Tool

Виберіть текст для аналізу

Історичний текст

Історичний текст
Kyiv was founded in the 5th century. In 882, Prince Oleg captured the city and made it the capital of Rus'.

Аналізувати

Результати аналізу

Сутність	Категорія	Важливість	Джерело
Kiev	Загальні знання	27%	Wikipedia
Oleg of Novgorod	Загальні знання	86%	Wikipedia
Kievan Rus'	Загальні знання	3%	Wikipedia

Рис. 8 Перегляд результатів

Text Analysis

Text to analyze

The Dnieper is the longest river in Ukraine. It originates in Russia, flows through Belarus, and empties into the Black Sea.

Analyze

Analysis Results

Entity	Source	Link	Confidence
Dnieper	Wikipedia	View	30,0%
Ukraine	Wikipedia	View	24,0%
Russian reversal	Wikipedia	View	77,0%
Belarus	Wikipedia	View	21,0%
Black Sea	Wikipedia	View	89,0%

Рис. 9 Перегляд результатів

Named Entity Recognition

Text to analyze

The Dnieper is the longest river in Ukraine. It originates in Russia, flows through Belarus, and empties into the Black Sea.

Analyze

Analysis Results

Text	Category	Subcategory	Confidence
Dnieper	Location	Немає	99,0%
river	Location	Немає	69,0%
Ukraine	Location	CountryRegion	100,0%
Russia	Location	CountryRegion	100,0%
Belarus	Location	CountryRegion	95,0%
Black Sea	Location	Немає	99,0%

Рис. 9 Перегляд результатів

Висновки

Додаток успішно інтегрує Azure Text Analytics для ефективного аналізу тексту, що дозволяє отримувати точні результати без розробки власних NLP-алгоритмів. Завдяки використанню ASP.NET Core та DI-контейнера архітектура є гнучкою і масштабованою. Інтерфейс, реалізований на Razor Pages, є зручним для користувача. Програма має широке практичне застосування — від аналізу новин до обробки наукових текстів і документів. Завдяки моделі обробки помилок забезпечується надійна робота навіть при некоректних даних.