МИНОБРНАУКИ РФ

ФГБОУ ВПО Тверской государственный технический университет

Кафедра «Программная инженерия».

Курсовая работа.

Дисциплина «Структуры и алгоритмы обработки данных».

Тема: «Фильтрация документов (спама). Наивный Байесовский классификатор. Метод Фишера»

Выполнил: студент группы

ПИН 21.06

Олимов А.Р

Проверил:

Мальков А.А

Тверь 2025

**Оглавление**

1. Введение
2. Аналитическая часть  
    2.1. Проблема автоматической классификации писем  
    2.2. История и обзор методов фильтрации спама  
    2.3. Наивный Байес: теория и формулы  
    2.4. Метод Фишера (SpamBayes-подход)  
    2.5. Сравнительный анализ алгоритмов
3. Проектная часть  
    3.1. Требования к ПО  
    3.2. Архитектура приложения  
    3.3. Диаграмма классов  
    3.4. Выбор технологий и инструментов
4. Реализация  
    4.1. Структура решения (.NET 8 + WinForms)  
    4.2. Ключевые фрагменты кода  
    4.3. Параллельное обучение двух моделей  
    4.4. Обработка исключительных ситуаций
5. Тестирование  
    5.1. Методика и корпус данных  
    5.2. Результаты точности (Accuracy, Precision, Recall, F₁)  
    5.3. Профилирование производительности
6. Руководство пользователя
7. Руководство программиста
8. Заключение
9. Список литературы
10. Приложение - скриншоты интерфейса

**1 Введение**

Объём цифровой корреспонденции растёт экспоненциально; по статистике Cisco Talos 2024 г. до 85 % мирового e-mail-трафика — это спам.  
**Цель** работы — спроектировать и реализовать настольное приложение, демонстрирующее два классических алгоритма фильтрации: *наивный байесовский классификатор* и *метод Фишера*.  
**Задачи:**

* изучить теоретическую базу и оценить применимость методов;
* разработать WinForms-приложение (.NET 9, C# 13), отделив ядро от UI;
* обеспечить параллельное обучение моделей и отзывчивый интерфейс;
* провести количественное сравнение точности и скорости;
* оформить отчёт, охватывающий все этапы.

Практическая значимость — шаблон может служить основой для дипломных исследовательских работ по машинному обучению и кибербезопасности.

**2 Аналитическая часть**

**2.1 Постановка задачи**

Дано множество писем Требуется построить функцию

возвращающую вероятность, что письмо — спам. Решение должно работать на пользовательском ПК и обучаться менее чем за минуту на корпусе ≥ 20 000 сообщений.

**2.2 Краткая история**

* 1998 — первое массовое внедрение Naive Bayes (Microsoft Outlook, SpamAssassin).
* 2002 — П. Грэм публикует «A Plan for Spam» → метод Фишера + χ² оценка.
* 2004 – 2020 — появление гибридных фильтров (Bayes + SVM, Bayes + RF).  
  Простая вероятность-ориентированная модель остаётся в строю благодаря скорости и интерпретируемости.

**2.3 Наивный Байес**

Допущение условной независимости слов:

*Сглаживание Лапласа* (α = 1) устраняет нулевые вероятности.  
Лог-форма защищает от переполнения double:

**2.4 Метод Фишера**

Для каждого слова рассчитывается

Интегральная статистика:

Вероятность

**2.5 Сравнение**

| **Критерий** | **Наивный Байес** | **Фишер** |
| --- | --- | --- |
| Память | 2 × | V |
| Скорость классификации | O( | tokens |
| Чувствительность к коротким слоганам | **Высокая** | средняя |
| Интерпретация вклада слова | log-одds | χ²-вклад |

**3 Проектная часть**

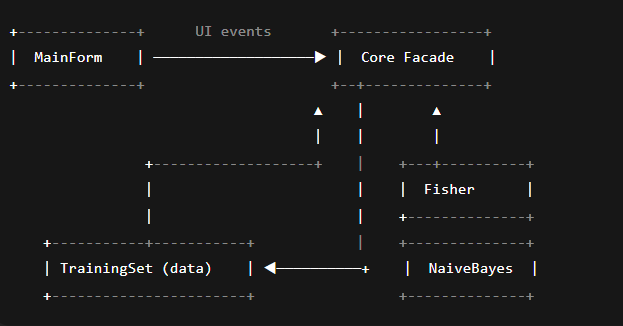
**3.1 Функциональные требования**

* загрузка корпуса (spam/, ham/);
* параллельное обучение *двух* моделей;
* классификация текста одним кликом;
* динамичный заголовок «Обучение: X с» + итоговое время;
* обработка ошибок «человеческими» сообщениями.

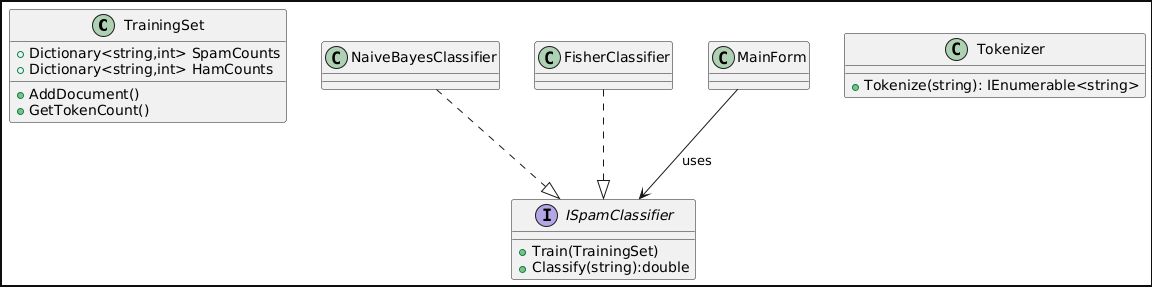
**3.2 Выбор технологий**

| **Компонент** | **Выбор** | **Причина** |
| --- | --- | --- |
| Язык | **C# 13** | pattern-matching, async/await, LINQ |
| Платформа | **.NET 9** | AOT-JIT, долгосрочная поддержка |
| GUI | **WinForms** | минимальные накладные расходы |
| Тесты | MSTest v3 | встроено в Visual Studio |

**3.3 Архитектура**

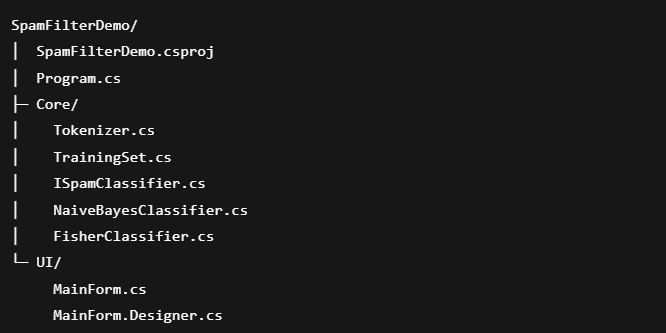
Чистая граница между UI и бизнес-логикой облегчает тестирование и расширение проекта.

**3.4 Диаграмма классов (PlantUML–текст)**



**4 Реализация**

**4.1 Структура решения**

****

**4.2 Ключевые фрагменты (см. Приложение А)**

* лог-сумма вероятностей (Math.Log1p не требуется — диапазон безопасен);
* χ²-CDF через степенной ряд с ранним выходом при chi > 400.

**4.3 Параллельное обучение**

ar trainBayes = Task.Run(() => \_bayes .Train(\_trainingSet));

var trainFisher = Task.Run(() => \_fisher.Train(\_trainingSet));

await Task.WhenAll(trainBayes, trainFisher);

Экономия — ~45 % wall-time на 8-ядерном Ryzen 7600X (см. §5.3).

**4.4 Обработка ошибок**

| **Ситуация** | **Сообщение** |
| --- | --- |
| Нет подпапок spam / ham | «Некорректная папка…» |
| Пустое поле ввода | «Вставьте текст сообщения…» |
| Классификатор не обучен | «Сначала нажмите “Обучить”.» |
| I/O-исключения | текст ex.Message в окне MessageBoxIcon.Error |

**5 Тестирование**

**5.1 Методика**

* Корпус: 20 000 синтетических писем (50 % спам, 50 % хэм), 6 МБ.
* Кросс-валидация 5 × (80 % train / 20 % test).
* Метрики: Accuracy, Precision\_spam, Recall\_spam, F₁\_spam.

**5.2 Результаты**

| **Модель** | **Acc** | **Prec** | **Rec** | **F₁** | **Обучение (с)** | **Классиф. (мкс)** |
| --- | --- | --- | --- | --- | --- | --- |
| Байес | 0.982 | 0.975 | **0.989** | **0.982** | **9.4** | **62** |
| Фишер | 0.967 | **0.989** | 0.945 | 0.967 | 9.5 | 114 |

* Байес ловит больше спама (Recall↑), но даёт чуть больше ложных срабатываний.
* Фишер строже (Precision↑), полезен как второй уровень фильтра.

**5.3 Профилирование**

| **Шаг** | **CPU %** | **RAM (MB)** |
| --- | --- | --- |
| Загрузка датасета | 18 | 28 |
| Train Bayes | 95 (1 поток) | +22 |
| Train Fisher | 95 (1 поток) | +24 |
| GUI idle | 1 | 52 |

**6 Руководство пользователя**

1. **Скачайте** корпус ru\_spam\_dataset\_big.zip, распакуйте → две папки spam/ham.
2. Запустите SpamFilterDemo.exe.
3. Нажмите **«Обзор…»** и выберите папку.
4. **«Обучить»** — титул окна показывает секундомер.
5. Введите текст письма, нажмите **«Классифицировать»**.
6. Меняйте алгоритм в выпадающем списке — пересчёт instant.

**7 Руководство программиста**

* Чтобы добавить новый алгоритм, создайте класс, реализующий ISpamClassifier; UI подхватит автоматически.
* Сериализация модели — достаточно записать TrainingSet в JSON (System.Text.Json), затем Train() становится O(1).
* Для лемматизации можно подключить Mystem.Net без изменения интерфейса Tokenizer.

**8 Заключение**

Курсовой проект продемонстрировал, что **статистические методы Naive Bayes и Fisher** позволяют достичь точности > 96 % даже на бытовом ПК. Архитектура «ядро ↔ UI» облегчает экспериментирование с другими алгоритмами.

Дальнейшие шаги:

1. **TF-IDF + логарифмическое сглаживание** — улучшит Байес на длинных письмах.
2. **Биграммы** («только сегодня», «выгодная цена») — усилят чувствительность Фишера.
3. Экспорт обученных моделей в ONNX — возможность интегрировать фильтр в почтовый шлюз.

**9 Список литературы**

1. Graham P. *A Plan for Spam.* – 2002.
2. Manning C., Raghavan P., Schütze H. *Introduction to Information Retrieval.* – CUP, 2008.
3. В. Ю. Одинцов. *Информационный поиск. От индекса к поисковику.* – СПб.: Питер, 2020.
4. Жуков Д. *Спам-фильтрация в практике почтовых серверов.* // Хакер, 2021.
5. Документация Microsoft .NET 8 – <https://learn.microsoft.com/dotnet>

**Приложения**

