
# Software project using .NET
# **n-gram search engine**

The goal of this project is to produce a search engine that allows indexing and searching textual web content based on short sequences of Unicode characters, where short means at least 4 characters from the Basic Multilingual Plane.

**Motivation:** Changing the way text should be encoded is potentially disruptive to existing corpus of text and users. This can happen in several ways, such as encoding new characters disunifying existing ones, deprecating existing characters, or discouraging certain sequences of characters. Yet, in some cases, such changes could bring significant improvements to the users, which need to be considered against risks of fracturing existing data and introducing security concerns.

Unfortunately, there is currently no way to assess how prevalent a given sequence of characters is on the web. Generic search engines preprocess text considerably, including normalizing it, ignoring case, white space, control characters, private use characters or even punctuation, and they do not index arbitrary strings in the middle of words.

This project aims to provide a tool to answer the question *how many instances of given sequence of characters exist on the web*? The index should keep track of a few example URLs containing the sequence but does not need to return them all if not practical and does not need to rank them.

**Team size:** 3–4 students

**Platform:** .NET, any supported language of choice

**Contact:** Jan Kučera (kucera@unicode.org)

**Example workload split:**

*Topic 1:* *Web crawling and processing input data*

An existing web crawler is expected to be utilized or adapted to access raw web data before processing. Plain text needs to be extracted from the data using correct encoding and avoiding any encoding-changing processing. This should be extensible based on the content type. For example, HTML and XML data should resolve character entities, JSON strings should be unescaped, etc. All the found n-grams need to be communicated to the storage.

***Possible extensions*:** support for more content types (PDF, ZIP, DOCX, etc.); distributed crawling.

*Topic 2:* *Database/index design and storage*

An appropriate storage system for the data needs to be designed, balancing space requirements and retrieval speed, while maximizing *n*. For example, small values of *n* could allow creating an index with constant lookup time. The index needs to provide the number of occurrences together with a few reference URLs per n-gram. However, a considerable number of URLs would likely need to be stored to ensure each contributes only once to the total number of occurrences.

***Possible extensions***: Store dates of crawl with the occurrence numbers, so that historical development over time can be reported; support characters not encoded yet.

*Topic 3:* *Web API and UI*

A web user interface and API to access and present the indexed data and statistics should be designed and documented. The interface should allow characters to be entered either directly or escaped and it should make clear which characters are entered, including control ones.

**Possible extensions:** Entering characters by their name; font selection; charts if history data is present.