Classification of Stellar Spectra using KMP and Levenshtein Distance Algorithm

Guntara Hambali - 13523114
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: guntarahambali3@gmail.com , 13523114@std.stei.itb.ac.id

***Abstract*— Star classification is a fundamental task in modern astronomy to understand the physical properties and evolution of stellar objects. Traditionally, classification relies on identifying absorption patterns in the stellar spectrum. Since those patterns appear in a sequence across wavelengths, it opens a possibility to approach the problem using string pattern matching. String matching is a computational method that identifies repeated or target sequences inside a larger sequence. In this paper, we implement exact and fuzzy string-matching algorithms, the Knuth-Morris-Pratt (KMP) and Levenshtein distance algorithms, to identify stellar types based on simplified spectral pattern. Using simulated data for A, G, and M stars, we test the both algorithms to classify certain star to a certain spectral type.**

***Keywords*—KMP Algorithm, Levenshtein Distance, Stellar Classification, Stellar Spectrum, String Matching**

I. Introduction

Star is one of the most fundamental astronomical objects observed throughout the history of astronomy. Understanding stars is a milestone to our understanding of the universe, as stars are major constituents of galaxies and provide clues about physical laws, time scales, and cosmic evolution. One of the most important properties of a star is its spectral type, which reveals its temperature, mass, and chemical composition. The classification of a star's spectral type is determined by analyzing its spectrum—the distribution of light intensity over different wavelengths—especially the presence and depth of certain absorption lines caused by various atomic and molecular species in the stellar atmosphere.

In recent decades, as astronomical observations have become more data-intensive, new approaches have been explored to automate spectral classification. Oneway to approach the problem is to treat the stellar spectrum as a sequence of values and apply string matching algorithms. This idea is based on the fact that absorption patterns form distinguishable sequences of intensity levels across wavelengths, which can be converted into symbolic representations. From this perspective, spectral classification can be reduced as a string pattern matching problem.

Two main algorithms are used in this paper: the Knuth-Morris-Pratt (KMP) algorithm for exact pattern matching and Levenshtein distance for fuzzy or approximate matching. The KMP algorithm is useful when the input spectrum is expected to closely match a known pattern, while Levenshtein distance is more tolerant to noise and minor shifts, which useful for realistic spectra where absorption features may not align perfectly.

The challenge in this inquiry is the nature of real spectral data. Public datasets such as those from the Sloan Digital Sky Survey (SDSS) or LAMOST contain millions of stellar spectra, but these are complex and often noisy. Furthermore, each spectrum is high-dimensional and may require significant preprocessing. For this reason, in this paper, we simulate simplified stellar spectra based on theoretical absorption lines found in A, G, and M-type stars. The simulated data serves as a controlled environment to test the effectiveness of string-matching methods for spectral classification.

This paper is organized as follows: Section 2 will explain the theoretical foundation of stellar spectra, including the physical characteristics of A, G, and M stars. It will also introduce the string-matching algorithms used in this paper. Section 3 will describe the implementation, beginning with the data simulation process, followed by the transformation of spectral data, and ends with algorithmic evaluation. In section 4, we will see the conclusion of this paper based on evaluation. Finally, Section 5 recommendations for future research.

II. Theoretical Basis

A. Stellar Spectra Definition

Stellar spectra is the distribution of a star's emitted light across different wavelengths. When starlight is passed through a spectrograph, it is decomposed into its constituent wavelengths, producing a spectrum that typically consists of a continuous background with dark absorption lines. These absorption lines are caused by the interaction of the stellar radiation with elements in the star's atmosphere, where specific wavelengths are absorbed due to electronic transitions in atoms or molecules. Stellar spectra are essential tools in astrophysics. They serve as "x-ray" of stars, revealing vital information such as chemical composition, surface temperature, radial velocity, surface gravity, and luminosity class of a star. Through analysis of spectral lines, astronomers can classify stars, study stellar evolution, and determine distances and motions within the galaxy.

B. Harvard Sprectral Classification System

Stellar spectra are traditionally classified into types according to the Harvard spectral classification system: O, B, A, F, G, K, and M. These types are ordered from hottest (O) to coolest (M), and the classification is primarily based on the strength and nature of certain absorption lines, particularly hydrogen and various metal lines. The characteristic of each class is described in the table below:

Table 1. Characteristic of each spectral class

Spectral Type	Temperature Range (K)	Dominant Features
O	> 30,000	Ionized helium lines
B	10,000 – 30,000	Neutral helium and strong H
A	7,500 – 10,000	Strong hydrogen Balmer lines
F	6,000 – 7,500	Weaker H lines, Ca II strong
G	5,200 – 6,000	Metal lines (Ca, Fe), H weak
K	3,700 – 5,200	Strong metal and molecular bands
M	< 3,700	Molecular bands (TiO, VO)

This study focuses specifically on spectral types A, G, and M due to the distinctive and contrasting spectral features they have. A-type stars, with surface temperatures between 7,500 and 10,000 K, are known for their strong and well-defined hydrogen Balmer absorption lines, such as Hα (656.3 nm), Hβ (486.1 nm), and Hγ (434.0 nm). These features make them particularly suitable for pattern recognition using string matching algorithms, as the absorption patterns are sharp and consistent. G-type stars, with temperatures between 5,200 and 6,000 K, represent solar-like stars and are characterized by a combination of weak hydrogen lines and prominent metal lines, especially from calcium (e.g., Ca II H and K lines near 393 and 397 nm) and iron. This moderate complexity makes G-type spectra ideal for evaluating algorithms on mixed-feature patterns. In contrast, M-type stars have surface temperatures below 3,700 K and exhibit spectra dominated by complex molecular absorption, primarily from TiO bands. The spectral patterns of M-type stars are dense and irregular, providing a challenging test case for matching algorithms [2]. By selecting these three types—A, G, and M—the study spans a broad range of spectral behaviors and complexity, allowing a comprehensive evaluation of string-matching techniques applied to stellar spectral data.

In reality, there are always noises in the spectrum data. In general, the cause comes from random signals during the conversion process from analog signals to digital signals. In addition, extrinsic factors can also come from changes in thermal signals in conductive materials on the instrument that can affect the detector's work in capturing. The second factor is very dependent on the instrument used. Fortunately, both factors have a typical distribution pattern, namely the Gaussian distribution. [1]

C. Knuth-Morris-Pratt Algorithm

The Knuth–Morris–Pratt (KMP) algorithm is an string matching algorithm designed to search for the occurrence of a pattern string (P) within a larger text string (T) in linear time. Traditional brute-force matching algorithms often re-examine characters in the text that have already been compared, leading to potentially quadratic time complexity in the worst case. In contrast, KMP avoids redundant comparisons by precomputing a prefix table (also called the border function), which encodes the longest proper prefix of the pattern that is also a suffix. This prefix table is used to determine the next position in the pattern to resume comparison after a mismatch, allowing the algorithm to shift the pattern efficiently without rechecking previously matched characters.

The KMP algorithm operates in two main phases: preprocessing and searching. In the preprocessing phase, the prefix table is constructed for the given pattern in O(m) time, where m is the length of the pattern. This table helps identify how far the pattern can safely shift after a mismatch. In the searching phase, the algorithm compares the pattern with the text from left to right. When a mismatch occurs, the prefix table is consulted to skip ahead in the pattern without restarting the comparison from the beginning. The overall time complexity of the algorithm is O(n+m), where n is the length of the text and m is the length of the pattern, making it highly suitable for matching long sequences. [3]

In the context of this study, the KMP algorithm is applied to pattern matching within stellar spectra that have been converted into string representations. Specifically, each stellar spectrum, which originally a sequence of intensity values over wavelengths, is normalized and transformed into a string through symbolic binning or scaling. This string can then be processed as textual data. Characteristic spectral features, such as absorption patterns unique to A, G, or M-type stars, are extracted as template patterns. Using KMP, the algorithm searches for these template patterns within a given input spectrum string to determine whether a particular stellar type is present. Since the KMP algorithm is deterministic and exact, it is particularly effective when matching high-contrast absorption patterns such as the Balmer lines in A-type stars. Its efficiency and precision make it well-suited for this simulation-based study, where the goal is to detect the presence or absence of distinctive spectral features with minimal computational overhead.

D. Levenshtein Algorithm

The Levenshtein distance algorithm is a technique used to quantify the difference between two strings by measuring the minimum number of edit operations required to transform one string into another. These operations typically include insertion, deletion, and substitution of single characters. [4] Unlike exact string-matching algorithms such as Knuth–Morris–Pratt (KMP), which only detect perfect matches, the Levenshtein distance allows for inexact matches by accounting

for small variations or noise within the data. The algorithm operates in O(nm) time, where n and m are the lengths of the two strings, by constructing a matrix that records the optimal edit distance between every prefix of the source and target strings. This characteristic makes Levenshtein particularly valuable in scenarios where the input data may be distorted, incomplete, or affected by random noise.

In the context of this study, where simulated stellar spectra are converted into discrete strings for classification, the Levenshtein distance is used to measure similarity between a target spectrum and known spectral templates of types A, G, and M. Because stellar spectra, even when simulated, may exhibit slight variations in intensity, line width, or line position due to instrumental or observational factors, relying solely on exact matches (as in KMP) could lead to misclassification. Levenshtein distance, on the other hand, enables the algorithm to tolerate small shifts, distortions, or noise in the spectral pattern while still recognizing its overall structure. This flexibility makes it advantageous for comparing complex or irregular patterns such as those found in M-type spectra, which often contain overlapping molecular bands. While the computational cost of Levenshtein is higher than KMP, its ability to quantify similarity rather than demand perfection provides a more robust and adaptable solution for the task of identifying stellar types based on spectral pattern matching. This approach enhances the accuracy of classification in situations where the spectrum may not be ideally clean or when comparing different stars within the same spectral class.

E. Spectrum Data Representation

To apply string matching algorithms to stellar spectra, it is first necessary to convert the analog spectral data into a digital format that can be processed algorithmically. A stellar spectrum is originally recorded as a continuous function representing light intensity as a function of wavelength, typically across the visible spectrum range of approximately 400–700 nanometers. This data is sampled into discrete points to form an array of intensity values, each corresponding to a specific wavelength interval. In order to utilize string-based algorithms such as Knuth–Morris–Pratt (KMP) and Levenshtein distance, these numerical sequences must be transformed into symbolic strings.

There are two principal approaches to digital representation: scaling to digits and symbolic binning. In the first approach, intensity values are multiplied by a constant scaling factor (e.g., 100) and rounded to the nearest integer, resulting in a sequence of numeric characters (e.g., 0.86 → "86"). These digit sequences are then concatenated into a single string, preserving relative intensity patterns while enabling character-wise comparison. The second approach involves binning the intensity values into qualitative categories—such as ‘A’, ‘B’, ‘C’, and ‘D’—based on predefined intensity thresholds. This symbolic representation reduces sensitivity to minor fluctuations and is especially useful for handling noise or variation in line depth and width. For instance, high-intensity regions may be labeled ‘A’, while deep absorption lines may be categorized as ‘D’, creating a discrete symbolic pattern that captures the general shape of the spectrum. In this paper, we will use the second method because it is more robust to tolerate the noise or redshift.

This transformation process enables stellar spectra to be treated as sequences of characters, which can then be compared using classical string-matching algorithms. It preserves the key structural features of the spectrum such as the position and depth of absorption lines while abstracting away the precise numerical values that are often affected by observational noise.

III. Implementation and Result Analysis

A. Problem Mapping

To implements their spectrum, we must understand that stellar spectra are composed of:

- Wavelength (λ): A sequence of values, usually in nanometers (nm), representing the position of light in the electromagnetic spectrum (e.g., 400–700 nm for visible light).
- Intensity (I): A normalized value (between 0 and 1) that represents how much light is received at each wavelength. Absorption lines appear as dips in this intensity.

Each star type (e.g., A, G, M) has characteristic absorption lines, defined by:

- Center wavelength (λ_0): The position of the line.
- Amplitude: How deep the line dips below the continuum.
- Width: How broad the line spreads around λ_0.

Our goal is to simulate this data. After simulating the spectral data as continuous intensity values over a range of wavelengths, the next crucial step is to transform this numerical data into symbolic form. This transformation is necessary because string matching algorithms such as KMP and Levenshtein distance operate on sequences of discrete characters.

To achieve this, the intensity values of the spectrum are discretized into bins, where each bin represents a range of normalized intensity values. The idea is rooted in the observation that absorption features appear as intensity dips in the spectrum, and these dips can be categorized by their depth. Last, we can then apply the algorithm into the transformed data.

B. Data Simulation

Before applying string matching algorithms to classify stellar types based on their spectra, we must first have the spectral data that reflects the properties of different types of stars. Unfortunately, obtaining a comprehensive set of real observational spectra that covers a wide and controlled variety of stellar types presents several challenges. While large-scale surveys such as SDSS and LAMOST have produced millions of stellar spectra, these datasets are still limited in terms of their representativeness for algorithmic experimentation. Real spectra vary in resolution, are affected by redshift and instrumental noise, and often contain incomplete or

ambiguous features due to observational constraints. More importantly, observational datasets rarely provide clean, ideal examples of each spectral class under uniform conditions, which makes them unsuitable as a baseline for evaluating algorithmic pattern recognition.

There are thus two main difficulties in using real-world spectral data for this purpose. First, the data is often too large and complex to be processed directly using lightweight symbolic algorithms without heavy preprocessing or dimensionality reduction. Second, it is not structured in a way that supports controlled comparisons, since real spectra introduce noise, shifting features, and overlapping characteristics that make it difficult to isolate specific spectral types for evaluation. Fortunately, stellar spectra are governed by well-understood physical laws which can be used to generate synthetic spectra that represent real observations in idealized conditions. These simulations can be scaled and structured precisely to reflect the defining features of A, G, and M-type stars as described in the theoretical section.

Therefore, in this inquiry, we simulate the spectral data using models that reflect the typical absorption behavior of the chosen stellar types. This enables us to create a dataset that is both representative of real stellar spectra and sufficiently simplified for algorithmic analysis, without the complications introduced by observational variability.

To generate synthetic stellar spectra that resemble real observations, this study constructs each spectrum as a normalized continuum with absorption features. These features are mathematically modeled using Gaussian profiles, which approximate the shape and behavior of spectral absorption lines found in stellar atmospheres. The function responsible for generating each individual absorption line is defined as follows:

```
def gaussian_absorption(wl, center,
amplitude, width):
    return 1 - amplitude * np.exp(-0.5 *
((wl - center) / width) ** 2))
```

This function takes as input an array of wavelengths (wl), the central wavelength of the line (center), the amplitude (how deep the absorption line dips), and the width (how broadly the feature spreads). It returns an intensity curve where a Gaussian-shaped dip is subtracted from a baseline intensity of 1, to simulate how light is absorbed at specific wavelengths in a star's atmosphere.

The full spectrum of a given star type is generated using the following function:

```
def simulate_star_spectrum(wl_range,
lines, noise=0.005):
      wl = np.linspace(*wl_range, 1000)
      spectrum = np.ones_like(wl)
      for line in lines:
          center, amp, width = line
          spectrum *=
gaussian_absorption(wl, center, amp,
width)
      spectrum += np.random.normal(0,
noise, wl.shape)
    return wl, spectrum
```

This function begins by creating an evenly spaced wavelength array wl (which its value when called will be from 400 to 700 to represent visible light spectrum). A flat continuum (spectrum) is initialized with intensity values of 1.0 across all wavelengths. Then, for each absorption line defined in the lines list, the gaussian_absorption function is applied multiplicatively to the spectrum. This models the cumulative effect of all absorption lines on the continuum. Finally, Gaussian noise is added to each intensity value, simulating the random variations and imperfections commonly observed in real astronomical instruments.

Then, based on theoretical basis, there are unique criteria for each spectrum class, which represent the chemical contents of a star.

```
  a_type_lines = [
      (656.3, 0.4, 1),
      (486.1, 0.3, 1.2),
      (434.0, 0.2, 1.5)
  ]
  g_type_lines = [
      (589.0, 0.15, 0.8),
      (517.0, 0.1, 0.6),
      (486.1, 0.1, 1.2)
  ]
  m_type_lines = [
      (705.0, 0.2, 2),
      (620.0, 0.15, 2),
      (575.0, 0.15, 1.5)
]
```

Using all functions made above, we then call the simulate_star_spectrum function

```
wl_range = (400, 700)

wl, spec_A =
simulate_star_spectrum(wl_range,
a_type_lines)
_, spec_G =
simulate_star_spectrum(wl_range,
g_type_lines)

_, spec_M =
simulate_star_spectrum(wl_range,
m_type_lines)
```

The simulated data can be visualized as:

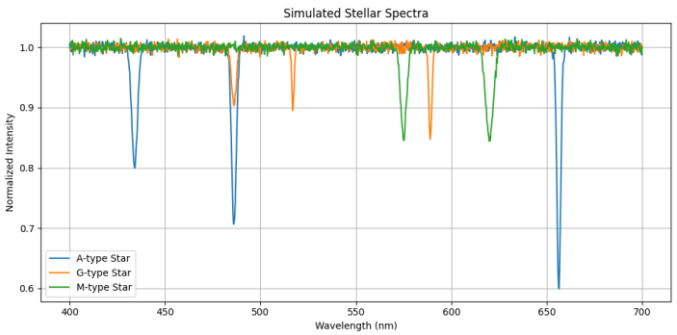

Fig 1. Simulated Stellar Spectra

C. *Spectral Data Transformation*

To apply string matching to the spectral data, we should transform the data. This technique, often referred to as symbolic representation, converts continuous spectral data into a string composed of discrete symbols representing varying levels of absorption. We implement this as follows:

```
def
stringify_spectrum_bins(intensity_array):
      def bin_intensity(i):
          if i > 0.9: return 'A'
          elif i > 0.8: return 'B'
          elif i > 0.7: return 'C'
          else: return 'D'

      return ''.join(bin_intensity(i) for
i in intensity_array)
```

The function defines bin_intensity(i), a nested classification function that maps each intensity value i to a symbol: 'A' for high intensity (i.e., minimal absorption), 'B' for mild absorption, 'C' for moderate absorption, and 'D' for deep absorption. Specifically, the binning thresholds are defined as follows: 'A' for values above 0.9, 'B' for 0.8–0.9, 'C' for 0.7–0.8, and 'D' for values less than or equal to 0.7. These threshold values is a heuristic discretization of the normalized intensity scale, where 1.0 represents the continuum (no absorption) and values closer to 0 indicate stronger line absorption. The function processes the entire intensity_array.

D. *Knuth-Morris-Pratt Algorithm Implementation*

After transforming the spectral data, we then can implement the Knuth-Morris-Pratt Algorithm. First, we generate the border function:

```
def generate_border_function(pattern):
  border_function = [0] * len(pattern)
  length = 0
  i = 1
  while i < len(pattern):
   if pattern[i] == pattern[length]:
     length += 1
     border_function[i] = length
     i += 1
   else:
     if length != 0:
        length = border_function[length -
1]
     else:
        border_function[i] = 0
       i += 1
  return border_function
```

Then, we implement the KMP algorithm like this:

```
def kmp_search(text, pattern):
  border_function=
generate_border_function(pattern)
  i = j = 0
  while i < len(text):
   if text[i] == pattern[j]:
     i += 1
     j += 1
   if j == len(pattern):
     return i - j
   elif i < len(text) and text[i] !=
pattern[j]:
     if j != 0:
       j = border_function[j - 1]
     else:
       i += 1
  return -1
```

Last, we can use use the the kmp_search we made before as follows:

```
input_str=
transform_spectrum(crop_spectrum(wl,
spec_A, 480, 10))

pattern_str=
transform_spectrum(crop_spectrum(wl,
spec_A, 486.1, 1))

print("Input  :", input_str)
print("Pattern:", pattern_str)

found= kmp_search(input_str, pattern_str
)
if found != -1:
    print("Pattern found at index", found)
else:
    print("Pattern not found."
```

The result is:

```
Input:
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAABBBCCDCCCCBBAAAAAAA

Pattern: CCDCCCC

Pattern found at index 50
```

E. Levenshtein Algorithm Implementation

We then now can apply the Levenshtein algorithm to search (fuzzy match) the pattern of G spectral type stellar in the simulated data of A spectral type stellar.

```
_, spec_G = simulate_star_spectrum((400,
700), g_type_lines)
pattern_G =
transform_spectrum(crop_spectrum(wl,
spec_G, 486.1))

input_str = transform_spectrum(snippet)

score = levenshtein(input_str, pattern_G)

print("Similarity to G-type:", score)
```

The output is as follows:

```
Similarity to G-type: 0.33962264150943394
```

It means that the simulated data is not really similar to G spectral type stellar which as we expected. Now, we use the same way but now the simulated data is of G spectral type stellar. The outpus shows as follows:

```
Similarity to G-type: 0.7547169811320755
```

It means that the simulated data is similar to G spectral type stellar which we expected.

IV. Conclusion

From the implementation of the string-matching strategy, the KMP algorithm successfully identified the target spectral pattern CCDCCCC which represents the absorption profile of a G-type star at index 50 within the symbolic spectrum of a simulated G-type star. This confirms that exact pattern matching can detect distinct spectral fingerprints under ideal conditions. For a more tolerant evaluation, the Levenshtein distance algorithm was applied. When comparing the symbolic spectrum of an A-type star to the G-type pattern, the resulting similarity was only 33.96%, whereas the G-type star's similarity to the same pattern reached 75.47%. This significant difference confirms the capability of Levenshtein algorithm to distinguish between spectral types even in the presence of noise and overlapping features. Therefore, it can be concluded that this inquiry successfully demonstrates the feasibility of using string matching algorithms for stellar spectral classification. Further development and evaluation with more diverse patterns and real data will be discussed in the next section.

V. Recommendation

Based on the simulated spectra, string matching algorithms such as KMP and Levenshtein distance were able to distinguish stellar spectral types by identifying characteristic absorption patterns. It will be a development in further research if we apply this method to real spectral data from astronomical surveys such as SDSS or LAMOST. That research would serve as a verification of this paper—whether the symbolic matching strategy remains valid under real observational noise and spectral resolution. After all, using real spectra will require additional steps, especially in preprocessing and normalization to convert raw data into symbolic form. Further research on optimizing the stringification process or learning the symbolic bins automatically from data would also be a valuable recommendation for future refinement.

VI. Closing Statement

The author would like to thank the entire academic community who have organized the Strategi Algoritma lectures for the 2024/2025 Informatics Engineering class of the Bandung Institute of Technology. Especially to the lecturer, Mr. Rinaldi Munir and all the assistants of the Innovation and Computational Engineering (IRK) laboratory.

The author realizes that there are still many shortcomings in this study. Therefore, this study is very open to all kinds of constructive criticism and suggestions. The author is also open to any intention to collaborate in developing this research.

References

[1] W. D. Pence, R. L. White, and A. S. Greenfield, "Lossless Astronomical Image Compression and the Effects of Noise," *Publications of the Astronomical Society of the Pacific*, vol. 121, no. 877, pp. 441–458, 2009, doi: 10.1086/599023.

[2] "Harvard Spectral Classification," *Swinburne University of Technology* [Online]. Available: https://www.geeksforgeeks.org/dsa/introduction-to-levenshtein-distance/ . [Accessed on: Jun. 24, 2025, 19:30].

[3] Munir, Rinaldi. (2025), "Pencocokan string (string matching) dengan algoritma brute force, KMP, Boyer-Moore)". [Accessed on: Jun. 24, 2025, 19:00].

[4] "Introduction to Levenshtein distance," *geeksforgeeks*[Online]. Available: https://astronomy.swin.edu.au/cosmos/h/harvard+spectral+classification .[Accessed on: Jun. 24, 2025, 19:20].

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 24 Juni 2025

Guntara Hambali - 13523114