**A\* Algoritması ile 8 Taş Oyun Tasarımı ve Uygulaması**

**RAPORU**

**Hazırlayan**

Fırat Kaan Bitmez

**Öğrenci Numarası**

23281855

**Dersin Hocası**

Doç.Dr. Gökhan Kayhan

**Giriş**  
A\* algoritması, yol bulma problemlerini çözmek için kullanılan etkili bir arama algoritmasıdır. Ancak, 8 taş oyunu gibi durum tabanlı oyunlar için kullanmak uygun olmayabilir çünkü bu tür oyunlar genellikle çok daha karmaşık bir durum uzayına sahiptir. Fakat biz Yapay Zeka dersi kapsamında 8 taş oyununu A yıldız algoritması ile oluşturup anlık olarak durumunu izleyebileceğimiz bir tasarım ve kodlama yapacağız.

**Kullanılan Kütüphaneler ve Araçlar**

**NLTK** (Natural Language Toolkit): Metin normalizasyonunda yaygın olarak kullanılan bir NLP kütüphanesi. NLTK kütüphanesini dahil etmek için Terminal komutuyla “**Pip intall nltk“** bilgisayarımıza dahil ediyoruz. Daha sonra ihtiyacımız olan diğer modülleri kodlamanın başlangıcında import ederek projeye dahil edeceğiz.

**NLTK(**wordnet**): Wordnet** NLTK kütüphanesine bağlı bir dil veritabanıdır. Wordnet kelime dağarcıklarını ve bu kelimeler arasındaki ilişkileri içeren bir veritabanıdır.

**NLTK(**punkt**): Punkt** NLTK’nin cümle ve kelime düzeyinde metinleri bölme(token) için kullandığı veri kaynağını indirir.

**string**: Python dilinde bulunan string işlemleri için yerleşik kütüphane.

**re** (Regular Expressions): Düzenli ifadelerin kullanılmasını sağlayan Python modülü.

**Kullanılan Normalizasyon Teknikleri**

**Case Normalization** (Büyük/Küçük Harf Normalleştirme): Metin içindeki tüm karakterlerin küçük harfe dönüştürülmesi.

**Punctuation Removal** (Noktalama İşaretlerinin Kaldırılması): Metinden noktalama işaretlerinin kaldırılması.

**Stop Word Removal** (Durak Kelime Kaldırma): Metinden yaygın durak kelimelerin (stop words) kaldırılması.

**Stemming** (Kök Çıkarma): Kelimelerin köklerinin çıkarılması.

**Lemmatization**: Kelimelerin sözlükteki köklerine dönüştürülmesi.

**Tokenization** (Belirteçleme): Metnin kelimelere veya belirteçlere bölünmesi.

**Synonyms and Abbreviation Replacement** (Eşanlamlılar ve Kısaltmaların Değiştirilmesi): Metindeki eşanlamlıların ve kısaltmaların tam hâllerine dönüştürülmesi.

**Removing Numbers and Symbols** (Sayıların ve Sembollerin Kaldırılması): Metinden sayıların ve sembollerin kaldırılması.

**Removing Remaining Non-Textual Elements** (Kalan Metin Dışı Unsurların Kaldırılması): Metindeki kalan HTML etiketleri veya URL'lerin kaldırılması.

**Kodlama**

#Bu Proje FIRAT KAAN BİTMEZ tarafından hazırlanmıştır.

#Projede Amacımız 9 farklı normalizasyon tekniklerini öğrenerek bir text normalization uygulaması yazmaktır.

# pip install nltk komutuyla termail üzerinden kütüphanemizi yükeldikten sonra

#nltk kütüphanesinden gerekli modülleri import ediyoruz

from nltk.stem import PorterStemmer, WordNetLemmatizer

import string

import re

import nltk

import nltk

nltk.download('wordnet')

import nltk

nltk.download('punkt')

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

# 1) Case Normalization (Büyük/Küçük Harf Normalleştirme)

#Case normalization'da amacımız bütün harfleri küçük hale getirmek

text\_case = "The quick BROWN Fox Jumps OVER the lazy dog."

text\_case = text\_case.lower()

Bu kodlama sonucunda aldığımız Çıktı:

the quick brown fox jumps over the lazy dog.

# 2) Punctuation Removal (Noktalama İşaretlerinin Kaldırılması)

#Noktalama işaretlerinin kaldırılması, metinden özel karakterleri ve noktalama işaretlerini kaldırma

text\_punctuation = "The quick BROWN Fox Jumps OVER the lazy dog!!!"

text\_punctuation = text\_punctuation.translate(str.maketrans("", "", string.punctuation))

Bu kodlama sonucunda aldığımız Çıktı:

The quick BROWN Fox Jumps OVER the lazy dog

# 3) Stop Word Removal (Durak Kelime Kaldırma)

#Durak kelime kaldırma, “the” ve “a” gibi anlamı çok az olan yaygın kelimelerin kaldırılmasıdır.

nltk.download('stopwords')

text\_stopwords = "The quick BROWN Fox Jumps OVER the lazy dog."

stop\_words = set(stopwords.words("english"))

words = text\_stopwords.split()

filtered\_words = [word for word in words if word.lower() not in stop\_words]

text\_stopwords = " ".join(filtered\_words)

Bu kodlama sonucunda aldığımız Çıktı:

quick BROWN Fox Jumps lazy dog.

# 4) Stemming (Kök Çıkarma)

#kelimeleri kök haline getirme işlemidir

stemmer = PorterStemmer()

text\_stemming = "running,runner,ran"

words = text\_stemming.split(",")

stemmed\_words = [stemmer.stem(word) for word in words]

text\_stemming = ",".join(stemmed\_words)

Bu kodlama sonucunda aldığımız Çıktı:

run,runner,ran

# 5) Lemmatization

#Lemmatizasyon, “koşuyor” kelimesini “koş” olarak kök haline getirirken kelimenin kullanıldığı bağlamı

#dikkate alarak kelimeleri kök haline getirme işlemidir

lemmatizer = WordNetLemmatizer()

text\_lemmatization = "running,runner,ran"

lemmatized\_words = [lemmatizer.lemmatize(word, pos='v') for word in text\_lemmatization.split(",")]

text\_lemmatization = ",".join(lemmatized\_words)

Bu kodlama sonucunda aldığımız Çıktı:

run,runner,run

# 6) Tokenization (Belirteçleme)

#Belirteçleme, metni bireysel kelimeler veya ifadeler olarak ayırmak anlamına gelir

text\_tokenization = "The quick BROWN Fox Jumps OVER the lazy dog."

text\_tokenization = re.sub(r'[^\w\s]', '', text\_tokenization)

tokens = word\_tokenize(text\_tokenization)

Bu kodlama sonucunda aldığımız Çıktı:

['The', 'quick', 'BROWN', 'Fox', 'Jumps', 'OVER', 'the', 'lazy', 'dog']

# 7) Replacing synonyms and Abbreviation to their full form to normalize the text in NLP (Eşanlamlıları ve Kısaltmaları Tam Hâllerine Dönüştürme)

#Bu teknik, metin verisindeki eşanlamlıları veya kısaltmaları tam hâllerine dönüştürmek gerektiğinde faydalıdır.

text\_synonyms = "I'll be there at 2pm"

synonyms = {"I'll": "I will", "2pm": "2 pm"}

for key, value in synonyms.items():

    text\_synonyms = text\_synonyms.replace(key, value)

Bu kodlama sonucunda aldığımız Çıktı:

I will be there at 2 pm

# 8) Removing numbers and symbol to normalize the text in NLP (Sayıların ve Sembollerin Kaldırılması)

#Bu teknik, metin verilerinde önemli olmayan sayıları ve sembolleri kaldırmak gerektiğinde faydalıdır.

text\_numbers = "I have 2 apples and 1 orange #fruits"

text\_numbers = re.sub(r"[\d#]", "", text\_numbers)

Bu kodlama sonucunda aldığımız Çıktı:

I have apples and orange fruits

# 9) Removing any remaining non-textual elements to normalize the text in NLP (Kalan Metin Dışı Unsurların Kaldırılması)

#Bu teknik, metin verilerinde HTML etiketleri, URL'ler ve e-posta adresleri gibi metin dışı unsurları

#kaldırmak gerektiğinde faydalıdır.

text\_non\_textual = "Please visit <a href='www.example.com'>example.com</a> for more information or contact me at info@example.com"

text\_non\_textual = re.sub(r"(<[^>]+>)|(http[s]?://(?:[a-zA-Z]|[0-9]|[$-\_@.&+]|[!\*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+)", "", text\_non\_textual)

Bu kodlama sonucunda aldığımız Çıktı:

Please visit example.com for more information or contact me at info@example.com

# 10) Birleştirilmiş Normalizasyon Fonksiyonu

def normalize\_text(text):

    text = text.lower()

    text = text.translate(str.maketrans("", "", string.punctuation))

    nltk.download('stopwords')

    stop\_words = set(stopwords.words("english"))

    words = text.split()

    filtered\_words = [word for word in words if word.lower() not in stop\_words]

    text = " ".join(filtered\_words)

    stemmer = PorterStemmer()

    stemmed\_words = [stemmer.stem(word) for word in text.split()]

    text = " ".join(stemmed\_words)

    lemmatizer = WordNetLemmatizer()

    lemmatized\_words = [lemmatizer.lemmatize(word, pos='v') for word in text.split()]

    text = " ".join(lemmatized\_words)

    text = re.sub(r'[^\w\s]', '', text)

    tokens = word\_tokenize(text)

    synonyms = {"I'll": "I will", "2pm": "2 pm"}

    for key, value in synonyms.items():

        text = text.replace(key, value)

    text = re.sub(r"[\d#]", "", text)

    text = re.sub(r"(<[^>]+>)|(http[s]?://(?:[a-zA-Z]|[0-9]|[$-\_@.&+]|[!\*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+)", "", text)

    return text

# Test metni

text = "The quick BROWN Fox Jumps OVER the lazy dog. I'll be there at 2pm. Please visit <a href='www.example.com'>example.com</a> for more information or contact me at [info@example.com](mailto:info@example.com)"

# Tüm normalizasyon tekniklerini içeren fonksiyonun uygulanması

normalized\_text = normalize\_text(text)

Bu kodlama sonucunda aldığımız Çıktı:

quick brown fox jump lazi dog ill pm pleas visit hrefwwwexamplecomexamplecoma inform contact infoexamplecom

# Normalizasyon yöntemlerinin çıktıları

print("Case Normalization (Büyük/Küçük Harf Normalleştirme):")

print(text\_case)

print("\nPunctuation Removal (Noktalama İşaretlerinin Kaldırılması):")

print(text\_punctuation)

print("\nStop Word Removal (Durak Kelime Kaldırma):")

print(text\_stopwords)

print("\nStemming (Kök Çıkarma):")

print(text\_stemming)

print("\nLemmatization:")

print(text\_lemmatization)

print("\nTokenization (Belirteçleme):")

print(tokens)

print("\nReplacing synonyms and Abbreviation to their full form to normalize the text in NLP (Eşanlamlıları ve Kısaltmaları Tam Hâllerine Dönüştürme):")

print(text\_synonyms)

print("\nRemoving numbers and symbol to normalize the text in NLP (Sayıların ve Sembollerin Kaldırılması):")

print(text\_numbers)

print("\nRemoving any remaining non-textual elements to normalize the text in NLP (Kalan Metin Dışı Unsurların Kaldırılması):")

print(text\_non\_textual)

print("\nBirleştirilmiş Normalizasyon Fonksiyonu:")

print(normalized\_text)

**Tam Sayfa Terminal Çıktısı**

**metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Sonuç**

Bu Kodlama Raporunda Metinlerin Normalizasyon yapılabilmesi için 9 yöntem incelendi. Bu 9 yöntemin her biri için bir örnekle birlikte Python’da kodlarla birlikte test edildi. Her yöntem için Çıktı ile kontrol sağlandı. En sonda Ortak bir metinde 9 normalizasyon yöntemi birleştirilerek tek bir metne 9 normalizasyon işlemi yapılsa ne olur? Sorusuna cevap alabilmek amacıyla bir fonksiyon oluşturdu.

Bu çalışma ve kodlamalar sonucunda Text Normalization hakkında kısaca şunlar söylenebilir: NLP uygulamalarında temel bir ön işleme adımıdır ve metin verilerinin daha tutarlı ve işlenebilir bir formatta olmasını sağlar. Buradaki normalizasyon teknikleri, geniş bir metin işleme yelpazesinde kullanılabilir ve NLP projelerinde veri hazırlama aşamasında önemli bir rol oynar.