2

# Модуль побудови дидактичних зв’язків.

defmodule WikiAssistant.ConnectionsBuilder do

# Головна функція модулю.

# Аргумент source\_page містить цільову статтю. Аргумент related\_pages –  
 # зв’язані с нею статті.

def build\_connections(source\_page, related\_pages) do

source\_page = prepare\_page(source\_page)

related\_pages

|> Enum.map(fn related\_page ->

Task.async(fn ->

build\_connections\_with\_page(source\_page,   
 prepare\_page(related\_page))

end)

end)

|> Enum.map(&Task.await/1)

|> Enum.reject(&(is\_nil(&1.type) || &1.certainty\_factor == 0))

end

# Допоможна функція для побудови зв’язків.

defp build\_connections\_with\_page(source\_page, related\_page) do

source\_is\_parent\_cf = calclulate\_certainty\_factor(source\_page,   
 related\_page)

source\_is\_child\_cf = calclulate\_certainty\_factor(related\_page,   
 source\_page)

{connection\_type, final\_cf} =

cond do

source\_is\_parent\_cf == 0 && source\_is\_child\_cf == 0 ->

{nil, 0}

source\_is\_parent\_cf >= source\_is\_child\_cf ->

{:child, source\_is\_parent\_cf}

true ->

{:parent, source\_is\_child\_cf}

end

%{

page: related\_page.data["title"],

type: connection\_type,

certainty\_factor: final\_cf

}

end

# Фукнкція, що підраховує фактор впевненності у наявності зв’язку для двох   
 # статей.

defp calclulate\_certainty\_factor(parent\_page, child\_page) do

parent\_page

|> build\_facts(child\_page)

|> Enum.map(&(&1.certainty\_factor))

|> Enum.reduce(0, &((&1 + &2) - (&1 \* &2)))

end

3

# Функція, що виконує пошук фактів.

defp build\_facts(parent\_page, child\_page) do

fact\_lists = [

build\_intro\_link\_facts(parent\_page, child\_page),

build\_body\_link\_facts(parent\_page, child\_page),

build\_title\_inclusion\_facts(parent\_page, child\_page),

build\_content\_inclusion\_facts(parent\_page, child\_page)

]

Enum.flat\_map(fact\_lists, &(&1))

end

# Функція, що виконує пошук фактів першого типу.

def build\_intro\_link\_facts(parent\_page, child\_page) do

link\_exists? =

child\_page.extra.intro\_links

|> Enum.any?(&Enum.member?(parent\_page.extra.titles, &1["target"]))

if link\_exists?, do: [%{certainty\_factor: 0.4}], else: []

end

# Функція, що виконує пошук фактів другого типу.

defp build\_body\_link\_facts(parent\_page, child\_page) do

child\_page.extra.body\_links

|> Enum.filter(&Enum.member?(parent\_page.extra.titles, &1["target"]))

|> Enum.map(fn %{"position" => position} ->

factor = 0.2 \* (1 - position / child\_page.extra.body\_length)

%{certainty\_factor: factor}

end)

end

# Функція, що виконує пошук фактів третього типу.

defp build\_title\_inclusion\_facts(parent\_page, child\_page) do

title\_included? =

Enum.any? child\_page.extra.titles, fn child\_title ->

String.match?(child\_title, parent\_page.extra.titles\_regex)

end

if title\_included?, do: [%{certainty\_factor: 0.9}], else: []

end

# Функція, що виконує пошук фактів четвертого типу.

defp build\_content\_inclusion\_facts(parent\_page, child\_page) do

parent\_page.extra.title\_not\_inside\_link\_regex

|> Regex.scan(child\_page.extra.content, return: :index)

|> Enum.map(fn result ->

{pos, \_} = List.first(result)

%{certainty\_factor: 0.1 \* (1 - pos /   
 child\_page.extra.content\_length)}

end)

end

4

# Допоміжна функція, що виконує підготовку статті для побудови зв’язків.

defp prepare\_page(page) do

titles = [page["title"] | page["redirects"]]

escaped\_title = Regex.escape(page["title"])

escaped\_titles =

titles

|> Enum.map(&Regex.escape/1)

|> Enum.join("|")

titles\_regex = ~r/(#{escaped\_titles})/i

title\_not\_inside\_link\_regex = ~r/#{escaped\_title}(?![^\[\[]\*\]\])/i

intro\_text = page["sections"]["intro"]["text"]

intro\_length = String.length(intro\_text)

intro\_links = page\_section\_links(page, "intro")

body\_text = page["sections"]["body"]["text"]

body\_length = String.length(body\_text)

body\_links = page\_section\_links(page, "body")

content = intro\_text <> body\_text

content\_length = intro\_length + body\_length

%{

data: page,

extra: %{

titles: titles,

titles\_regex: titles\_regex,

title\_not\_inside\_link\_regex: title\_not\_inside\_link\_regex,

intro\_links: intro\_links,

body\_length: body\_length,

body\_links: body\_links,

content: content,

content\_length: content\_length

}

}

end

# Допоміжна функція, що збирає посилання з конкретної секції статті.

defp page\_section\_links(page, section) do

Enum.map page["sections"][section]["links"], fn link ->

%{

"target" => link["target"] |> String.split("#") |> List.first,

"position" => link["position"]

}

end

end

end