Jan Proniewicz, 297989, Informatyka - Data Science

Języki i Biblioteki Analizy Danych - Projekt zaliczeniowy, pt. “Kategoryzacja notatek prasowych / artykułów internetowych”

Ogólny opis systemu:

Omawiany projekt działa na zasadzie aplikacji konsolowej. Tzn. że nie posiada graficznego interfejsu i wszystkie funkcjonalności odbywają się za pośrednictwem konsoli. Działanie systemu testowano za pośrednictwem środowiska PyCharm, ale powinien również być możliwy do uruchomienia w wierszu poleceń (należy wtedy uruchomić plik *Menu.py*). Na potrzeby zajęć, powstał on w języku Python (autor korzystał z wersji 3.6).

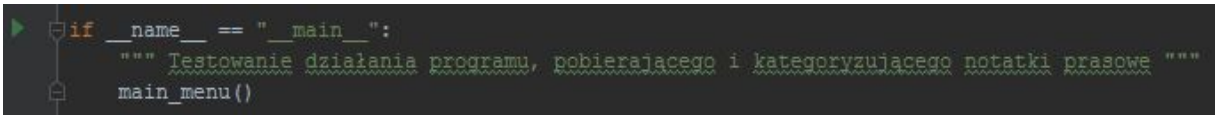
Celem niniejszego programu jest umożliwienie użytkownikowi pobranie z internetu “najnowszych” notatek prasowych lub artykułów (dla uproszczenia, pojęcia te będą stosowane zamiennie), ich kategoryzację, względem odgórnie określonych kategorii oraz ich szybki przegląd. Źródłem tych notatek są wybrane przez autora strony internetowe portali prasowych. W tym przypadku, są to Wyborcza (*wyborcza.pl*), Onet (*wiadomosci.onet.pl* i *sport.onet.pl*) oraz Wirtualna Polska (*wiadomosci.wp.pl* i *sportowefakty.wp.pl*). Informacje są zbierane z różnych zakładek tych stron internetowych, np. świat, kraj, Kraków, Śląsk itd.. Oczywiście, każdy z tych portali ma swój własny zestaw zakładek. Ograniczono się tylko do niektórych z nich. Poprzez “najnowsze” artykuły, autor ma na myśli te, które są dostępne na pierwszej stronie portalu (pierwsza stronica, bez rozwijania) oraz takie, które nie zostały opublikowane wcześniej, niż określonego dnia (o takiej dacie “granicznej” decyduje użytkownik; dokładniej to zostanie wytłumaczone później).

W ramach niniejszego systemu, każda notatka zawiera: tytuł artykułu, krótki opis na wstęp (tzw. lead), datę publikacji, odsyłacz URL do oryginalnego artykułu oraz nazwę zakładki, z której ta notatka pochodzi. Pobrane notatki są później zapisywane do

pliku *press_notes.json*. Dzięki temu, pobrane ówcześnie dane nie zostaną utracone, a program będzie miał do nich dostęp nawet po ponownym uruchomieniu.
Jak już wspomniano wcześniej, kategoryzacja pobranych notatek prasowych opiera się na ich przydziale do odgórnie określonych przez system kategorii, tj. “polityka”, “sport”, “piłka nożna”, "zimowy sport", "tenis", "sport motorowy", "kolarstwo", “koronawirus”, “medycyna”, “nauka i technika”, “gospodarka”, “kultura”, “religia i Kościół”, “kryminalne i wypadki” i “inne” (kategoria domyślna, w której znajdują się artykuły, niepasujące do żadnej z wymienionych grup). Aplikacja nie wyklucza sytuacji, w której jedna notatka prasowa mogłaby się znaleźć w różnych kategoriach. Omawiany proces jest dokonywany na podstawie zakładki, z której dana notatka pochodzi oraz treści, wynikającej z tytułu oraz lead’u. Wyniki klasyfikacji zostają później zapisane w pliku *notes_classified.json*, co usuwa konieczność ponownej kategoryzacji, jeśli program zostałby zamknięty i później - uruchomiony jeszcze raz. Warto również zaznaczyć, że artykuły w obu wymienionych plikach są posortowane wg daty publikacji, w kolejności od najnowszego do najstarszego, w celu ułatwienia ich czytelności.
Posegregowane artykuły można wyświetlić w konsoli, co umożliwia specjalnie do tego utworzone menu. W czytelny sposób pokazuje pobrane informacje na ich temat, tj. tytuł, lead, data publikacji oraz odsyłacz do pełnej treści oryginalnego artykułu. Ze względu na charakter aplikacji (tj. konsola), notatki są wypisywane w kolejności od najstarszej do najnowszej, żeby użytkownik nie musiał “przewijać” okna konsoli na samą górę, by zobaczyć najnowsze wiadomości.
System pozwala również na usunięcie z “bazy”, tj. plików *.json*, starych notatek prasowych, w celu zwolnienia pamięci. Należy jednak pamiętać, iż jest to proces nieodwracalny.

Przewodnik użytkownika:

Żeby uruchomić niniejszą aplikację, należy wywołać w wierszu poleceń plik *Menu.py* lub, jeżeli użytkownik korzysta z PyCharm’a, kliknąć zielony trójkąt obok *if __name_ == “__main__”* w tym samym pliku, jak na rysunku poniżej.



Po chwili, konsola powinna wyświetlić “menu główne”, w którym są wylistowane możliwe funkcjonalności systemu.

```
C:\Python\python.exe C:/Users/Jan/Documents/python/JanProniewicz_Projekt/program_files/Menu.py
=============================================================================================================
System klasyfikacyjny internetowych notatek prasowych:
=============================================================================================
Wybierz opcję:
1 - Pobierz więcej notatek z internetu (strony: wyborcza.pl, onet.pl, wp.pl);
2 - Usuń 'stare' notatki prasowe;
3 - Przeprowadź klasyfikację notatek prasowych;
4 - Pokaż notatki prasowe wg kategorii;
5 - Wyjdź
```

Użytkownik może pobrać najnowsze artykuły ze wszystkich trzech podanych portali prasowych, usunąć stare notatki, sklasyfikować pobrane notatki, wyświetlić ich spis wg kategorii lub po prostu wyjść. Wybór opcji następuje poprzez wpisanie w konsoli odpowiedniej liczby.

Chcąc pobrać więcej notatek prasowych, system w pierwszej kolejności poprosi użytkownika o podanie roku, miesiąca i daty (liczbowo). Będą one reprezentowały tzw. “minimalną datę”, która będzie stanowić punkt przerwania pobierania, tzn. program nie będzie brał pod uwagę artykułów opublikowanych wcześniej, niż tego dnia. Po zakończeniu procesu, powinna się wyświetlić liczba notatek, pobrana z każdego portalu (przy czym uwzględnia wszystkie notatki w bazie, a nie pobrane w danym momencie) oraz stosowny komunikat. Niestety, ze względu na złożoność algorytmu, funkcja ta działa bardzo wolno (szczególnie jeżeli postanowi się pobrać nowe notatki po bardzo długim czasie od ostatniego użytkowania), dlatego wykonując ją, należy uzbroić się w cierpliwość. Użytkownik ostatecznie powinien mieć widok taki, jak na rysunku.

```
=============================================================================================
Podaj 'najwcześniejszy' dzień (pełna data), do którego notatki będą pobierane:
Rok: 2021
Miesiąc: 1
Dzień: 5
Liczba notatek prasowych w bazie:
Wyborcza: 585
Onet: 886
Wp: 755
Razem: 2226
Pobieranie zakończone pomyślnie!
```

Usuwanie po stronie użytkownika opiera się na podobnej zasadzie, tzn. proces prosi o podanie roku, miesiąca i dnia. Tym razem, oznaczają one “datę maksymalną”, stanowiącą punkt przerwania usuwania. Program usunie tylko te artykuły, które

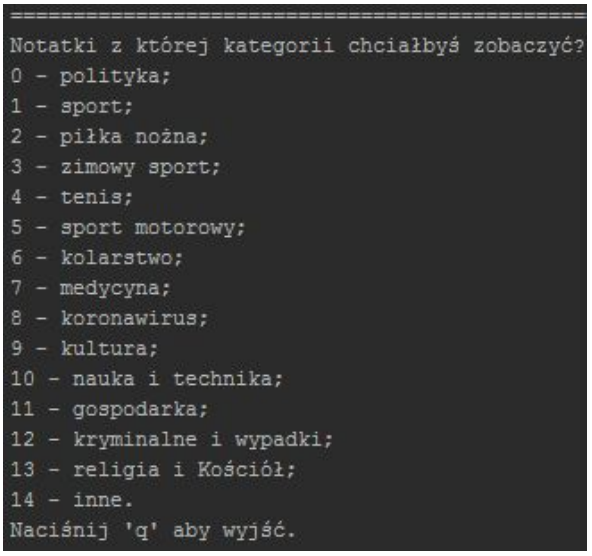
zostały opublikowane przed danym dniem. Po wykonaniu funkcji, system wyświetli liczbę usuniętych notatek oraz stosowny komunikat, jak na rycinie poniżej.

```
================================================================================
Podaj 'najpóźniejszy' dzień (pełna data), do którego notatki będą usuwane:
UWAGA! Proces nieodwracalny!
Rok: 2021
Miesiąc: 1
Dzień: 5
Usunięto: 28
Usuwanie 'starych' notatek prasowych zakończone pomyślnie!
```

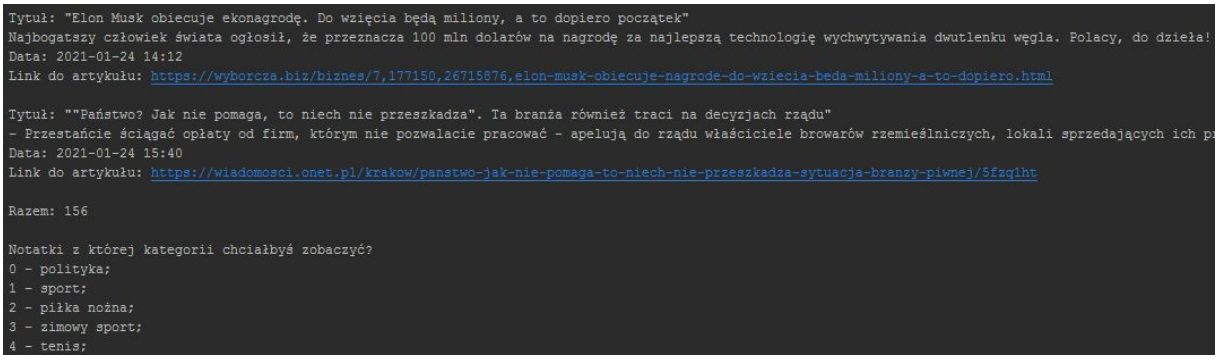
Klasyfikacja będzie wymagała od użytkownika podjęcia jednej decyzji. Mianowicie, czy użytkownik będzie chciał zresetować zbiór sklasyfikowanych notatek, tj. opróżnić plik *notes_clssified.json* i sklasyfikować wszystko od nowa (*y* na “Tak”), czy tylko podzielić nowo pobrane notatki, a “te stare zostawić w spokoju” (*n* na “Nie”). Pierwsza metoda pozwala na rekonfigurację “bazy” skategoryzowanych artykułów, jeśli program przeszedł przez znaczące zmiany, a druga - jest w pewnym sensie bardziej oszczędna. Mimo to, ponownie, ze względu na postać algorytmu, funkcja ta może działać stosunkowo wolno. Po zakończeniu procesu, system wyświetla liczbę notatek prasowych w każdej z kategorii, jak poniżej.

```
================================================
Czy chcesz skategoryzować notatki od nowa?
Tak (y) / Nie (n)y
Liczba artykułów/notatek w każdej z kategorii:
Polityka: 332
Sport: 904
Piłka nożna: 306
Zimowy sport: 262
Tenis: 146
Sport motorowy: 127
Kolarstwo: 29
Medycyna: 270
Koronawirus: 447
Kultura: 94
Nauka i technika: 95
Gospodarka: 156
Kryminalne i wypadki: 303
Religia i kościół: 33
Inne: 235
Kategoryzacja notatek prasowych zakończona pomyślnie!
```

Po wyborze opcji wyświetlania posegregowanych notatek prasowych, system przekieruje użytkownika do osobnego menu, w którym ma po kolei wylistowane kategorie i odpowiadające im indeksy. Żeby wyświetlić artykuły dla któreś z nich, należy wprowadzić odpowiedni indeks. Można również wprowadzić ‘q’, aby wrócić do menu głównego.



Jeśli użytkownik zdecyduje się na coś, program wypisuje wszystkie notatki z danej kategorii w kolejności od najstarszej do najnowszej, w celu ułatwienia czytelności na konsoli. Dzięki temu, klient będzie mógł z łatwością zobaczyć aktualne wiadomości z danej dziedziny (ogólne informacje, wynikające z tytułu i krótkiego opisu). Co więcej, jeśli jakiś temat go zainteresuje, może również kliknąć odpowiedni odsyłacz, który przekieruje go w przeglądarce do pełnego artykułu (Wyborcza niestety wymaga wykupienia prenumeraty, żeby mieć pełny dostęp do treści artykułów). Następnie, system wraca do menu kategorii. Poniżej przedstawiono widok (jego fragment) końca wizualizacji listy dla notatek prasowych z kategorii “gospodarka”.



Jeżeli w jakimkolwiek momencie użytkownik poda dane, które system nie rozpozna, np. losowy ciąg znaków, wyświetli on stosowny komunikat o błędzie i wróci do menu głównego. Można to uznać za sposób anulowania operacji. Należy mieć na uwadze, że w przypadku opcji 1 i 2 system zareaguje dopiero wtedy, gdy wszystkie parametry dla daty zostaną wypełnione (prawidłowo lub nie). Aby zakończyć działanie aplikacji, wystarczy w menu głównym podać cyfrę 5.

Pełny opis systemu:

Zasadniczo, program, realizujący wymienione zadania, dzieli się na sześć plików programowych (*.py*) i dwa, reprezentujące “bazę” pobranych i posegregowanych notatek (*.json*). Pliki systemowe to:

- *Menu.py* - menu główne oraz plik inicjujący aplikację;
- *PressNotesDownloader.py* - klasa pobierająca notatki prasowe z Wyborczej, Onetu i WP;
- *PressNotesDeleter.py* - klasa usuwająca “stare” notatki prasowe;
- *PressNotesClassifier.py* - klasa kategoryzująca pobrane notatki prasowe;
- *PressNotesVisualizer.py* - klasa generująca w konsoli notatki prasowe z wybranych kategorii;
- *JsonOperations.py* - metody pomocnicze, pobierające treść z plików *.json* i zapisujące nowe informacje do nich.

Natomiast wśród plików “bazy” można wyróżnić:

- *press_notes.json* - tutaj są zapisywane pobrane notatki prasowe, z podziałem na portale, z których pochodzą, tj. “wyborcza”, “onet” i “wp”;
- *notes_classified.json* - jak sama nazwa wskazuje, jest to zapis skategoryzowanych notatek prasowych z podziałem na kategorie: “polityka”, “sport”, “piłka nożna”, "zimowy sport", "tenis", "sport motorowy", "kolarstwo", “koronawirus”, “medycyna”, “nauka i technika”, “gospodarka”, “kultura”, “religia i Kościół”, “kryminalne i wypadki” i “inne”.

Podzbiory notatek prasowych (dla portali i kategorii) są słownikami, w których każda notatka jest reprezentowana przez swój tytuł (jest on kluczem notatki). Z kolei same artykuły również są słownikami, o kluczach: “Tytuł”, “Opis” (lead), “Data”, “URL” i “Zakładka”.

Menu.py:

Samo menu opiera się na zasadzie pętli *while*, która nie zostanie przerwana, dopóki użytkownik nie zdecyduje się wyjść (opcja nr 5). Sprecyzowane zostały w niej instrukcje warunkowe, realizujące odpowiednie funkcje z pozostałych plików programowych, w zależności od *inputu* użytkownika. Obsługuje także wyjątki *ValueError* i *IndexError*, na wypadek, gdyby użytkownik wprowadził niezgodne dane w jakimkolwiek momencie. Wraca wtedy na początek pętli do menu głównego.

PressNotesDownloader.py:

Do pobierania informacji o notatkach prasowych, zastosowano dość niekonwencjonalną, aczkolwiek interesującą metodę, polegającą na tzw. *Web Scraping'u*. Polega ona na wydzielaniu określonych fragmentów kodu html wybranej strony. W pythonie, taką technikę umożliwia specjalna biblioteka o nazwie *beautifulsoup4*. Jej metoda *BeautifulSoup* potrafi zinterpretować zawartość strony internetowej, pobraną wcześniej np. przy pomocy biblioteki *requests* i przetworzyć ją na czytelny kod html. Następnie, dzięki innym metodom, wchodzących w skład omawianej biblioteki, można np. wyselekcjonować konkretne fragmenty tego kodu na podstawie typu i atrybutów, pobrać wartości tych atrybutów lub pobrać tekst, znajdujący się w danym "bloku". Żeby móc z łatwością wyselekcjonować fragmenty, których program rzeczywiście potrzebuje, warto na oryginalnej stronie internetowej wybrać opcję "Zbadaj" po wciśnięciu prawego przycisku myszy, aby wyświetlić jej kod html a następnie sprawdzić, które jego fragmenty dotyczą konkretnych elementów strony. Mając to na uwadze, można przejść do pewnego pisania algorytmu. Należy również pamiętać, że każda strona internetowa ma inną budowę, więc jeden program, stworzony dla jednej konkretnej strony może nie działać poprawnie dla wszystkich z nich. Dlatego, niezbędnym było utworzenie osobnych metod pobierania notatek prasowych dla Wyborczej, Onetu i WP. Dla dwóch ostatnich portali należało te metody jeszcze dodatkowo podzielić na funkcje wewnętrzne, ze względu na różny charakter niektórych zakładek / stron.

Sam proces pobierania notatek prasowych opiera się na utworzeniu obiektu klasy *PressNotesDownloader* i wykorzystaniu jej metod. W momencie jej utworzenia, zapisuje sobie wspomnianą "najwcześniejszą datę" (jej elementy są podawane jako argumenty) i przekształca ją na format *datetime* (do tego użyteczny jest moduł *datetime*). Oprócz tego, pobiera on zawartość pliku *press_notes.json*, aby móc później do niego zapisać nowo pobrane notatki oraz przygotowuje sobie spis zakładek dla każdego z portali (dla Wyborczej dodatkowo przygotowuje ciągi cyfr, będące charakterystycznymi częściami URL kolejnych zakładek).

Jak wspomniano wcześniej, klasa zawiera osobne metody, wykonujące Web Scraping dla poszczególnych portali:

- *scrape_wyborcza* - wykonująca Web Scraping dla Wyborczej (*wyborcza.pl*).

- *scrape_onet* - wykonująca Web Scraping dla Onetu, tj. *wiadomosci.onet.pl* i *sport.onet.pl*. Dzieli się na trzy osobne metody wewnętrzne. Jedna została stworzona dla stron, gdzie artykuły są wylistowane, druga - dla zakładek dot. miast i regionów (tam artykuły są ułożone jak “kafelki”), a trzecia z kolei pobiera niezbędne informacje z kolejnych artykułów.
- *scrape_wp* - wykonująca Web Scraping dla Wirtualnej Polski (*wiadomosci.wp.pl* i *sportowefakty.wp.pl*). Ze względu na całkowicie odmienny wygląd tych dwóch stron, dla każdej z nich utworzono osobne metody wewnętrzne.

Każda z wyżej wymienionych funkcji opiera się z grubsza na tych samych założeniach. Mianowicie, wykonują one odpowiednie operacje, iterując po kolejnych, przypisanych im kategoriach. W pierwszej kolejności, “budują” adres URL wybranej zakładki i za sprawą biblioteki *requests* - używają go do pobrania zawartości odpowiedniej strony internetowej. Następnie, przy pomocy *BeautifulSoup*, przetwarzają zawartość strony na kod html i wyszukują odpowiednich danych. M. in. wybierają “blok” kodu, zawierający informacje o artykułach/notatkach prasowych, dzielą go na poszczególne, osobne “bloczki” notatek, a potem, dla każdego z nich, wybierają niezbędne dane, opisane we wcześniejszych paragrafach, np. tytuł, lead itd. Wszystkie te informacje oraz nazwa przeglądanej zakładki są zbierane do kolejnych słowników, które następnie są dodawane do zbioru (słownika) artykułów z odpowiedniego źródła. Przy użyciu funkcji *save_press_notes* obiekt *PressNotesDownloader* zapisuje wszystko z powrotem do pliku *press_notes.json*.

Funkcje te obsługują również *AttributeError*, na wypadek, gdyby aktualnie przeglądana strona miała inny format, niż obsługiwany przez program (gdyż np. pochodzi z innego URL i *BeautifulSoup* nie mogłoby znaleźć szukanych elementów). Wtedy po prostu pomija dany artykuł i bada następny.

W trakcie pobierania notatek z internetu wykonywane są również procedury pomocnicze, np. weryfikacja, czy notatka o danym tytule się już pojawiła (jeśli tak, to najpewniej następne też będą się powtarzać, więc program przejdzie do analizy kolejnej zakładki), czy data publikacji nie jest “za stara” w porównaniu do ustalonej “minimalnej” oraz metoda sortująca pobrane już notatki prasowe wg daty publikacji od najnowszej do najstarszej (dla lepszej czytelności i zachowania spójności w algorytmach; tu przydatane są metody *datetime* i *OrderedDict* kolejno z modułów/bibliotek *datetime* i *collections*).

PressNotesDeleter.py:

W trakcie tworzenia obiektu klasy *PressNotesDeleter* program zapisuje w nim wspomnianą wcześniej “maksymalną datę” (jej elementy są podawane jako argumenty) oraz pobiera zawartość obu plików *.json*. Proces usuwania “starych” notatek obiera się na wywołaniu utworzonej metody *delete_old_press_notes*, która iteruje po kolejnych notatkach z każdego zbioru (dla konkretnych portali lub kategorii) od najstarszej do najnowszej. Jeżeli data publikacji artykułu jest wcześniejsza niż “maksymalna”, notatka zostanie usunięta. W przeciwnym wypadku, funkcja dobiega końca. Proces jest powtarzany każdego z dwóch plików .json (po zakończeniu operacji, można użyć *modified_press_notes*, aby zapisać nowy stan notatek prasowych do odpowiednich plików).

PressNotesClassifier.py:

Nietypowa sytuacja również prezentuje się w przypadku procesu kategoryzacji notatek prasowych. Ponieważ system umożliwia przydział notatki prasowej do kilku kategorii naraz, typowa klasyfikacja nie może zostać zastosowana. Dlatego, niniejszy algorytm opiera się na dwóch sposobach:

1. Przydział na podstawie zakładki, z której pochodzi dana notatka. W tym celu, program będzie sprawdzał wartość klucza “Zakładka” w każdej z notatek i sprawdzał, czy taka zakładka jednoznacznie wskazywałaby na jedną z ustalonych kategorii. Także, postanowiono, że jeżeli artykuł pochodzi z zakładki dla określonej dyscypliny sportu, to powinien być także w kategorii “sport”, a jeśli wystąpił w zakładce “koronawirus”, to znajdzie się również w “medycynie” (zazwyczaj notatki z zakładki “koronawirus” są ściśle powiązane z tematami medycznymi).
2. Przydział wg liczby wystąpień określonych form w tytule i opisie (lead’ie). Podczas pobierania informacji o notatkach prasowych nie brano pod uwagę pełnych treści artykułów, ponieważ byłoby to zbyt obciążające dla programu. Z drugiej strony, tytuł i lead powinne teoretycznie zawierać wszystkie najważniejsze informacje, związane z danym tematem, więc postanowiono oprzeć klasyfikację tylko na nich. Liczby wystąpień form obliczano za pomocą biblioteki *re* (*regex*), której metody potrafią wykryć obecności pewnych ciągów znaków (wyrażeń regularnych) w tekście. Żeby uniknąć błędnych

przyporządkowań notatek prasowych do niektórych kategorii, w związku z przypadkowymi wystąpieniami form, za “optymalną” liczbę wystąpień przyjęto ‘3’ (należy mieć na uwadze, że tytuły i lead’y są stosunkowo krótkie, więc zbyt duża liczba wymaganych wystąpień również może wpłynąć na jakość kategoryzacji). Jednakże, nie eliminuje to całkowicie błędów w klasyfikacji. Dla niektórych kategorii, takich jak “koronawirus”, czy dyscypliny sportowe, postawiono na sprawdzenie, czy chociaż jedno z wyrażeń kluczowych występuje (ponieważ są to tematyki bardziej specyficzne / węższe). Dodatkowo, jeśli program uzna, że artykuł powinien się znaleźć w kategorii dla jakiejś dyscypliny, to powinien być też w kategorii “sport” (system nie robi czegoś podobnego dla “koronawirusa”, gdyż samo wspomnienie o “pandemii” nie świadczy o tematyce medycznej). Jednakże, w związku z tym rośnie ryzyko, że w kategorii “sport” znajdą się artykuły, zupełnie nie związane z tą tematyką (bo np. w notatce przypadkowo pojawił się “rowerzysta”). Ale to prędzej wynika z braku wiedzy autora na temat sportu i powiązanych z nim słów (w szczególności nazwisk niektórych zawodników, drużyn czy wydarzeń).

Przy tworzeniu obiektu klasy *PressNotesClassifier*, system pobiera zawartość obu plików *.json* (z *press_notes.json* są “wyciągane” pobrane notatki, a do *notes_classified.json* będą one zapisywane po klasyfikacji). Także, przygotowuje listę ustalonych odgórnie kategorii tematycznych i odpowiadające im listy “wyrażeń klucz”. Są to fragmenty form określonych rzeczowników, czasowników itd., które zazwyczaj pozostają niezmienne przy odmianie przez przypadki, czasy itp.. Zawierają one także łańcuchy znaków z nawiasami kwadratowymi, wewnątrz których znajdują się określone litery. Biblioteka *re* interpretuje je jako wyrażenia regularne, gdzie litery w nawiasach są zamienne (np. “rann[aiy]” interpretuje jednocześnie jako “ranna”, “ranni” i “ranny”). Utworzona metoda *classify_press_notes* dla każdej badanej notatki tworzy pustą listę, w której będą umieszczane potencjalne kategorie dla niej. Jest ona uzupełniana za pośrednictwem dwóch wcześniej wspomnianych sposobów. Po zakończeniu “poszukiwań”, notatka jest wpisywana do wszystkich kategorii wymienionych w liście. Jeżeli nie odnaleziono żadnej kategorii dla artykułu, zostaje on umieszczony w grupie “inne”. Może to wynikać z istnienia potencjalnej kategorii, która nie została uwzględniona przez autora lub niedoboru kluczowych wyrażeń.

Należy pamiętać, że z daną dziedziną może być związanych wiele pojęć i nie będzie się pamiętało o wszystkich z nich od razu. Dlatego jest to jeden z tych algorytmów,

które warto jest cały czas kontrolować i uzupełniać o brakujące słowa klucze, w oparciu np. o artykuły z kategorii “inne” lub osoby bardziej zaznajomione z daną dziedziną.
Klasa zawiera również kilka metod pomocniczych, np. weryfikacja, czy notatka już należy do jednej z kategorii (jeśli tak, można założyć, że nstępne też już są zaklasyfikowane), sortowanie po dacie publikacji w kolejności malejącej (analogiczna do *PressNotesDownloader.py*), czy też reset klasyfikatora, opróżniającego plik *notes_classified.json*, aby móc przeprowadzić klasyfikację od nowa (przydatna, gdy algorytm zostanie np. uzupełniony o nowe słowa klucze lub kategorie). Rzecz jasna, istnieje również metoda odpowiedzialna za zapis zmian w klasyfikacji do odpowiedniego pliku.

PressNotesVisualizer.py

Obiekt klasy *PressNotesVisualizer* pobiera zawartość pliku *notes_classified.json* oraz określa listę wszystkich ustalonych kategorii. Jej główna metoda *continuous_visualization* odpowiada za tworzenie menu, przedstawionego w instrukcji użytkownika. Działa ono na zasadzie pętli, podobnie do *Menu.py* i podobnie jak tam, żeby wyjść, należy wprowadzić odpowiedni *input* (‘q’) (lub podać błędne dane, które menu główne potem obsłuży). Po wybraniu opcji przez użytkownika, program wywołuje metodę *show_press_notes_from_category*, wybierającą kategorię z listy, ściśle powiązaną z podanym wcześniej indeksem, a następnie wyświetla w odpowiednim formacie wszystkie notatki z tej kategori w kolejności od najstarszej do najnowszej.

Instalacja i podręcznik administratora:

Żeby program działał jak trzeba, potrzebne będą następujące biblioteki i moduły:

- *json* - pozwala operować na plikach typu *.json* (pobierać ich treść, modyfikować je);
- *requests* - pobiera zawartość stron internetowych na podstawie URL;
- *bs4 (beautifulsoup4)* - przekształca treść pobraną przez *requests* na html i pozwala na niej operować;
- *collections* - z niej pochodzi *OrderedDict*, w którym można sortować treść słowników;

- *datetime* - pomocna przy zamienianiu formatu daty ze *string* na *datetime* (bezpieczne rozwiązanie przy sortowaniu wg daty lub porównywaniu dat);
- *re (regex)* - pozwala operować na wyrażeniach regularnych.

Aby zainstalować program, należy najpierw pobrać plik *.zip*, dołączony wraz z niniejszą dokumentacją (lub pobrać z repozytorium, do którego link również został załączony w tym samym miejscu), a następnie wypakować jego zawartość w wybranym miejscu. Otrzymany folder warto otworzyć jako cały projekt np. w takim PyCharm'ie, ale uruchomienie skryptu *Menu.py* w wierszu poleceń też raczej powinno zadziałać. Przede wszystkim, nie należy nic zmieniać w układzie plików i folderów. W przeciwnym wypadku, aplikacja może nie zadziałać.
Także, jak już wspomniano wcześniej, procesy pobierania i klasyfikacji notatek prasowych mogą być czasochłonne. Zatem, korzystając z aplikacji, należy się uzbroić w cierpliwość.
Pomijając ewentualne pliki z folderów *.idea* i *__pycache__*, projekt dzieli się na dwa pakiety: pliki programowe z plikami *.py* (w tym główny plik uruchamiający) oraz pliki *.json*, stanowiące "bazę danych" programu.

Kod źródłowy:

W niniejszej części dokumentacji umieszczone zostały screeny, zawierające fragmenty kodu omawianej aplikacji (kod ten można również obejrzeć w samych plikach programowych). Obecność komentarzy powinna ułatwić zrozumienie sposobu działania algorytmów.

Menu.py:

Importowane pakiety:

```
# Jan Proniewicz, 297989, Informatyka - Data Science
""" Plik główny; menu obsługi konsolowej aplikacji """
# importowanie wszystkich niezbędnych narzędzi z elementów składowych systemu
from program_files.PressNotesDownloader import *
from program_files.PressNotesDeleter import *
from program_files.PressNotesClassifier import *
from program_files.PressNotesVisualizer import *
```

Kod główny:

```
def main_menu():
    """ Menu główne aplikacji; to za jego pośrednictwem użytkownik ma dostęp do funkcji systemu; znaki '=' pełnią tylko
     rolę 'estetyczną'/'separacyjną' (wizualnie oddzielają okna menu od siebie) """
    print("=============================================================================================================")
    print("System klasyfikacyjny internetowych notatek prasowych:")
    # "włączenie" systemu, działającego na zasadzie pętli
    all_systems_online = True
    while all_systems_online:
        # menu wychwytuje potencjalne błędy, tj. niepoprawne wartości wpisane przez użytkownika i wraca na początek pętli
        try:
            print("=============================================================================================================")
            # główne okno; wybór jednej z wypisanych opcji poprzez podanie w konsoli odpowiedniej cyfry
            option = input("Wybierz opcję:\n1 - Pobierz więcej notatek z internetu (strony: wyborcza.pl, onet.pl, wp.pl);\n"
                           "2 - Usuń 'stare' notatki prasowe;\n3 - Przeprowadź klasyfikację notatek prasowych;\n"
                           "4 - Pokaż notatki prasowe wg kategorii;\n5 - Wyjdź\n")
            # uruchamianie odpowiedniej funkcji systemu na podstawie wybranej opcji w menu; po zakończeniu wykonywania
            # wybranej z funkcji, program wraca na początek pętli (jeżeli użytkownik nie wybrał opcji '5')
            # pobranie nowych notatek prasowych
            if option == '1':
                print("=============================================================================================================")
                # podanie roku, miesiąca i dnia, reprezentujących 'granicę' pobierania notatek
                # (program nie będzie pobierał notatek wcześniejszych, niż sprecyzowana data)
                print("Podaj 'najwcześniejszy' dzień (pełna data), do którego notatki będą pobierane:")
                year = input("Rok: ")
                month = input("Miesiąc: ")
                day = input("Dzień: ")
                # utworzenie instancji obiektu 'PressNotesDownloader', odpowiedzialnej za pobieranie nowych notatek/
                # /artykułów z internetu (pobranie i zapis do pliku 'press_notes.json')
                pn_download = PressNotesDownloader(year, month, day)
                pn_download.get_more_press_notes()
                pn_download.save_press_notes()
                print("Pobieranie zakończone pomyślnie!\n")
            # usuwanie starych notatek prasowych / artykułów
            elif option == '2':
                print("=============================================================================================================")
                # podanie roku, miesiąca i dnia, reprezentujących 'granicę' usuwania notatek
                # (program zakończy pracę metody, jeżeli dotrze do notatek późniejszych, niż sprecyzowana data)
                print("Podaj 'najpóźniejszy' dzień (pełna data), do którego notatki będą usuwane:")
                print("UWAGA! Proces nieodwracalny!")
                year = input("Rok: ")
                month = input("Miesiąc: ")
                day = input("Dzień: ")
                # utworzenie instancji obiektu 'PressNotesDeleter', odpowiedzialnej za usuwanie starych notatek/
                # /artykułów z obu plików .json (tj. dla pobranych i sklasyfikowanych notatek)
                pn_delete = PressNotesDeleter(year, month, day)
                pn_delete.delete_old_press_notes()
                pn_delete.save_modified_press_notes()
                print("Usuwanie 'starych' notatek prasowych zakończone pomyślnie!\n")
            # klasyfikacja notatek prasowych dostępnych w bazie, tj. pliku 'press_notes.json'
            elif option == '3':
                print("=============================================================================================================")
                # użytkownik podejmuje decyzję, czy chce zresetować zawartość pliku 'notes_classified.json' i sklasyfikować
                # wszystkkie notatki od nowa, czy tylko nowo pobrane (musi wprowadzić jedną z dwóch odpowiednich liter)
                print("Czy chcesz skategoryzować notatki od nowa?")
                decision = input("Tak (y) / Nie (n)")
                if decision in ['y', 'n']:
                    # utworzenie instancji obiektu 'PressNotesClassifier', odpowiedzialnej za klasyfikację artykułów
                    pn_classify = PressNotesClassifier()
                    if decision == 'y':
                        # reset pliku 'notes_classified.json' z zaklasyfikowanymi notatkami
                        pn_classify.restart_classification()
                    # przeprowadzenie klasyfikacji i zapis wyników do pliku 'notes_classified.json'
                    pn_classify.classify_press_notes()
                    pn_classify.save_classification_changes()
                    print("Kategoryzacja notatek prasowych zakończona pomyślnie!\n")
                # jeśli 'input' jest nieprawidłowy, system wyrzuci wyjątek, który następnie zostanie obsłużony;
                # program wraca na początek pętli; analogiczna sytuacja występuje w pozostałych funkcjach, z tą różnicą,
                # że pojawienie się błędu wynika z właściwości zastosowanych metod
                else:
                    raise ValueError
            # wizualizacja posegregowanych notatek prasowych z pliku 'notes_classified.json'
            elif option == '4':
                print("=============================================================================================================")
                pn_vizualize = PressNotesVisualizer()
                pn_vizualize.continuous_visualization()
```

```
    # wyjście z aplikacji, tj. przerwanie petli
    elif option == '5':
        all_systems_online = False
    # jeśli 'input' opcji jest nieprawidłowy, system wyrzuci wyjątek, który następnie zostanie obsłużony
    else:
        raise ValueError
# obsługa wyjątków, tj. wyświetlenie komunikatu o błędzie i powrót na początek petli
except (ValueError, IndexError):
    print("Błąd wprowadzonych danych!\n")
```

JsonOperations.py:

Importowane pakiety:

```
# Jan Proniewicz, 297989, Informatyka - Data Science
""" Funkcje, operujące na plikach .json, tj. pobranie zawartości lub zapisanie danych do odpowiedniego pliku """
import json
```

Metody główne:

```
def get_data_from_json(json_file_name):
    """ Pobranie zawartości z wybranego pliku .json (json_file_name) """
    with open(json_file_name, "r", encoding="utf-8") as f:
        return json.load(f)
```

```
def set_data_to_json(json_file_name, data_to_set):
    """ Zapisanie danych (data_to_set) do wybranefo pliku .json (json_file_name) """
    with open(json_file_name, "w", encoding="utf-8") as f:
        json.dump(data_to_set, f, indent=4, ensure_ascii=False)
```

PressNotesDownloader.py:

Importowane pakiety:

```
# Jan Proniewicz, 297989, Informatyka - Data Science
""" Funkcja systemu, koncentrująca się na pobieraniu nowych notatek prasowych / artykułów z internetu;
 Wybrane platformy: wyborcza.pl, wiadomości.onet.pl, sport.onet.pl, wiadomości.wp.pl, sportowefakty.wp.pl """
import requests  # biblioteka, która posłuży do pobrania zawartości stron internetowych z podanego URL
from bs4 import BeautifulSoup  # metoda, która przetworzy pobraną zawartość na czytelny kod html i wydzieli jego odpowiednie części
from collections import OrderedDict  # import 'uporządkowanego słownika'
from datetime import datetime  # import niezbędnej metody z modułu 'datetime'
from program_files.JsonOperations import *  # import metod, operujących na plikach .json (metody własne)
```

Inicjalizacja obiektu klasy:

```
class PressNotesDownloader:
    """ Klasa 'PressNotesDownloader', która będzie pobierała notatki prasove z internetu, tj. ze strony 'Wyborczej',
    'Onetu' i 'Wirtualnej Polski' (wp)"""
    def __init__(self, min_year, min_month, min_day):
        """ Tworzenie instancji obiektu klasy 'PressNotesDownloader'; argumentami instancji są rok, miesiąc i dzień, tj.
         data, do której będą pobierane artykuły (późniejsze, niż ta data) """
        # uzupełnianie wartości dla miesiąca i dnia, jeśli użytkownik podał pojedyncze cyfry
        if len(str(min_month)) == 1:
            min_month = "0" + str(min_month)
        if len(str(min_day)) == 1:
            min_day = "0" + str(min_day)
        # utworzenie łańcuchowego formatu daty
        min_date_string = str(min_year) + "/" + str(min_month) + "/" + str(min_day)
```

```
# zastosowanie metody 'datetime' w celu przekształcenia daty na domyślny format datetime
# (rok-miesiąc-dzień godzina-minuty-sekundy); jeśli argumenty nie tworzą odpowiedniego formatu daty zgodnego z
# wybraną maską (%Y/%m/%d), program wyrzuca wyjątek, obsługiwany w głównym menu
self.min_date = datetime.strptime(min_date_string, '%Y/%m/%d')
# pobranie zawartości pliku 'press_notes.json', gdzie będą umieszczane artykuły po pobraniu
self.press_notes = get_data_from_json("..//json_files//press_notes.json")
# zakładki 'wyborcza.pl', z których będą pobierane artykuły
wyborcza_labels = ["swiat", "kraj", "sport", "zdrowie", "gospodarka", "technika", "nauka", "kultura"]
# lista ciągów cyfr, charakterystycznych dla URL poszczególnych zakładek
# (w tej samej kolejności co odpowiadające im zakładki)
html_id_for_wyborcza = ["0,75399", "0,75398", "0,154903", "TylkoZdrowie/0,0", "0,155287", "0,156282", "0,75400",
                        "0,75410"]
# uporządkowanie tych fragmentów URL dla zakładek Wyborczej w słowniku
self.wyborcza_categories = {}
for i in range(0, len(wyborcza_labels)):
    self.wyborcza_categories[wyborcza_labels[i]] = html_id_for_wyborcza[i]
# zakładki 'wiadomosci.onet.pl' i 'sport.onet.pl', z których będą pobierane artykuły
self.onet_labels = ["kraj", "swiat", "tylko-w-onecie", "religia", "pilka-nozna", "skoki-narciarskie", "formula-1",
                    "tenis", "kolarstwo", "krakow", "warszawa", "lodz", "wroclaw", "szczecin", "bialystok",
                    "poznan", "rzeszow", "trojmiasto", "slask", "opole"]
# zakładki 'wiadomosci.wp.pl' i 'sportowefakty.wp.pl', z których będą pobierane artykuły
self.wp_labels = ["koronawirus", "polska", "swiat", "spoleczenstwo", "polityka", "nauka", "slask", "krakow",
                  "najnowsze", "pilka-nozna", "zimowe", "tenis", "moto", "kolarstwo"]
```

Metody główne:

Pobieranie wiadomości z Wyborczej:

```
def scrape_wyborcza(self):
    """ Metoda wykonująca Web Scraping na stronie 'Wyborczej'; w innych słowach - metoda wychwytująca odpowiednie
    informacje o notatkach prasowych z kodu html 'Wyborczej' (tylko pierwsza stronica) """
    # pobranie aktualnego zbioru artykułów z Wyborczej
    wyborcza_news = self.press_notes["wyborcza"]
    # iteracja po kolejnych ustalonych zakładkach Wyborczej
    for cat in self.wyborcza_categories:
        # pobranie zawartości z internetu na podstawie URL
        url = "https://wyborcza.pl/" + self.wyborcza_categories[cat] + ".html"
        soup = self._get_soup(url)
        # wyodrębnienie odpowiedniego 'bloku' strony (blok zawierający spis artykułów na stronie)
        body = soup.find("div", {"class": "body"})
        # wyodrębnienie poszczególnych 'artykułów' (ich 'bloczków') z wcześniejszego bloku
        articles = body.find_all("li", {"class": ["entry even article", "entry odd article"]})
        # iteracja po wyodrębnionych notatkach prasowych
        for note in articles:
            # wyodrębnienie tytułu i linka do artykułu z html 'bloczka'; zakładka 'sport' ma nieco inny format niż
            # pozostałe
            if cat == "sport":
                title = note.find("h3").text.strip()
                art_url = note.find("h3").find("a").get("href")
            else:
                title = note.find("h2").text.strip()
                art_url = note.find("h2").find("a").get("href")
            # sprawdzenie, czy tytuł się powtarza wśród notatek Wyborzej
            title_repeats = self._check_if_title_repeats(wyborcza_news, title, cat)
            # jeśli tak, pętla jest przerywana i program bada następną zakładkę (gdyż następne artykuły w aktualnej
            # zakładce są już najpewniej 'stare', tj. dawno zapisane w 'bazie')
            if title_repeats == "YES":
                break
            # jeśli 'może', program przerywa iterację pętli i przechodzi do następnego artykułu
            elif title_repeats == "MAYBE":
                continue
            # pobranie zawartości z html 'bloczka' na temat lead'a i daty publikacji (format dzień-miesiąc-rok czas)
            lead = note.find("p", {"class": "lead"}).text.strip()
            original_date = note.find("span", {"class": "when"}).text
            # przekształcenie daty publikacji na ustalony format rok-miesiąc-dzień godzina:minuty
            art_date = self._modify_datetime_format(original_date)
            # sprawdzenie, czy artykuł nie jest 'za stary'; jeśli tak, pętla jest przerywana, a program bada
            # następną zakładkę
            if self.check_if_article_is_too_old(art_date):
                break
            # zapisanie pobranych danych o artykule w słowniku i umieszczenie ich w zbiorze notatek Wyborczej
            pressnote_info = {"Tytuł": title, "Opis": lead, "Data": art_date, "URL": art_url, "Zakładka": cat}
            wyborcza_news[title] = pressnote_info
    # posortowanie pobranych notatek w kolejności od najnowszej do najstarszej
    self.press_notes["wyborcza"] = self.sort_notes_by_date(wyborcza_news)
```

Pobieranie wiadomości z Onetu (z podziałem na zakładki “zwykłe” i “dla miast”):

Kod właściwy:

```
def scrape_onet(self):
    """ Metoda wychwytująca informacje o notatkach prasowych z kodu html Onetu """
    onet_news = self.press_notes["onet"]  # pobranie aktualnego zbioru artykułów z Onetu
```

```
# właściwa część metody 'scrape_onet'; iteracja po kolejnych ustalonych zakładkach Onetu
for cat in self.onet_labels:
    # zastosowanie odpoweidnich metod wewnętrznych w zależności od zakładki
    if cat in ["kraj", "swiat", "tylko-w-onecie", "religia", "pilka-nozna", "skoki-narciarskie", "formula-1",
               "tenis", "kolarstwo"]:
        scrape_onet_news_and_sport(cat)
    else:
        scrape_onet_regional(cat)
# posortowanie notatek z Onetu od najnowszej do najstarszej
self.press_notes["onet"] = self.sort_notes_by_date(onet_news)
```

Metoda wewnętrzna, zbierająca dane o artykule:

```
def create_press_note(art_url, category):
    """ Metoda wewnetrzna, pobierajaca i zapisujaca odpowiednie dane o notatce prasowej; art_url - link do
    aktualnego artykułu, category - aktualnie badana zakładka; True - wymuszenie pominięcia iteracji w petli
    'nadrzednej' (continue), False - przerwanie petli 'nadrzednej' (break) """
    # jeśli artykuł nie pochodzi z jednej z niżej podanych stron iteracja jest pomijana (zabezpieczenie przed
    # linkami dla reklam lub artykułami o nieodpowiednim formacie)
    if "wiadomosci.onet.pl" not in str(art_url) and "sport.onet.pl" not in str(art_url):
        return True
    # pobranie zawartości ze strony artykułu w postaci kodu html
    soup2 = self._get_soup(art_url)
    # pobranie informacji dot. tytułu notatki prasowej
    title = soup2.find("h1", {"class": "mainTitle"}).text.strip()
    # sprawdzenie, czy notatka się powtarza w zbiorze artykułów z Onetu
    title_repeats = self._check_if_title_repeats(onet_news, title, category)
    if title_repeats == "YES":
        return False  # przerwanie petli i przejście do następnej zakładki
    elif title_repeats == "MAYBE":
        return True  # pominięcie iteracji i przejście do następnego artykułu
    # pobranie informacji o lead'ie i dacie publikacji artykułu
    lead = soup2.find("div", {"id": "lead"}).text.strip()
    art_date = soup2.find("meta", {"itemprop": "datePublished"}).get("content")[:16]  # potrzebne jest tylko 16
    # pierwszych znaków (rok-miesiąc-dzień godzina:minuty (maska: 0000-00-00 00:00))
    # sprawdzenie, czy notatka nie jest 'za stara'; analogicznie do 'scrape_wyborcza'
    if self.check_if_article_is_too_old(art_date):
        return False
    # zapisanie zdobytych danych do słownika i umieszczenie powstałej notatki w zbiorze artykułów Onetu
    pressnote_info = {"Tytuł": title, "Opis": lead, "Data": art_date, "URL": art_url, "Zakladka": category}
    onet_news[title] = pressnote_info
    return True
```

Metoda wewnętrzna, pobierająca notatki prasowe z zakładek “zwykłych”:

```
def scrape_onet_news_and_sport(category):
    """ Metoda wewnetrzna, wykonująca web scraping na 'głównych' zakładkach 'wiadomosci.onet.pl' (nie dotyczących
    konkretnych miast) oraz 'sport.onet.pl'; artykuły na tych zakładkach są 'wylistowane', podobnie do #yborczej;
    category - aktualnie badana zakładka """
    # przygotowanie URL dla aktualnej zakładki
    if category in ["kraj", "swiat", "tylko-w-onecie", "religia"]:
        url = "https://wiadomosci.onet.pl/" + category
        if category == "religia":
            url = url + "/aktualnosci"
    else:
        url = "https://sport.onet.pl/"
        if category == "skoki-narciarskie":
            url = url + "zimowe/"
        url = url + category
    # pobranie zawartości zakładki w postaci kodu html
    soup = self._get_soup(url)
    # wyselekcjonowanie 'bloku' html, poświęconego artykułom
    stream_section = soup.find("section", {"class": "stream"})
    # wyselekcjonowanie poszczególnych 'bloczków', odpowiadających kolejnym notatkom prasowym
    articles = stream_section.find_all("div", {"class": "listItem listItemSolr itarticle"})
```

```
    # iteracja po kolejnych notatkach
    for note in articles:
        # wychwycenie i obsługa wyjątku, jeżeli element, który program próbował wyodrębnić, nie istniał
        # (w związku z nieodpowiednim 'niestandardowym' formatem strony artykułu); pominięcie iteracji
        try:
            # pobranie linka do oryginalnego artykułu z 'bloczka';
            # ostatni z możliwych elementów typu 'anchor' w html bloczka
            art_url = note.find_all("a")[-1].get("href")
            # dalsze operacje na html samego artykułu (w tym pobranie reszty informacji o notatce) oraz podjęcie
            # decyzji o postąpieniu z petlą (patrz.: komentarze w 'create_press_note')
            should_we_keep_going = create_press_note(art_url, category)
            if should_we_keep_going:
                continue
            else:
                break
        except AttributeError:
            continue
```

Metoda wewnętrzna, pobierająca notatki prasowe z zakładek “dla miast”:

```
def scrape_onet_regional(city):
    """ Metoda wewnetrzna, wykonująca web scraping na zakładkach 'wiadomosci.onet.pl', dot. wybranych miast;
    artykuły na tych zakładkach są ułożone jak 'kafelki'; city - aktualnie badane miasto/zakładka """
    # przygotowanie url zakładki i pobranie zawartości jej strony
    url = "https://wiadomosci.onet.pl/" + city
    soup = self._get_soup(url)
    # wyodrebnienie 'bloku' z poszukiwanymi artykułami
    items_section = soup.find("div", {"class": "items solrList"})
    # wyodrebnienie poszczególnych 'kafelków' (ich elementów typu 'anchor'), odpowiadających kolejnym artykułom
    article_anchors = items_section.find_all("a", {"class": ["itemBox itemBoxBig itemBoxLast",
                                                             "itemBox itemBoxNormal",
                                                             "itemBox itemBoxNormal itemBoxLast"]})
    # iteracja po kolejnych notatkach
    for anchor in article_anchors:
        # obsługa wyjątku na wypadek natrafienia na stronę z innym formatem
        try:
            # pobranie linka do oryginalnego artykułu i wykorzystanie go do dalszej analizy
            art_url = anchor.get("href")
            should_we_keep_going = create_press_note(art_url, city)
            # podjęcie decyzji o postąpieniu z petlą (patrz.: komentarze w 'create_press_note')
            if should_we_keep_going:
                continue
            else:
                break
        except AttributeError:
            continue
```

Pobieranie wiadomości z WP:

Kod właściwy:

```
def scrape_wp(self):
    """ Metoda wychwytująca informacje o notatkach prasowych z kodu html Wirtualnej Polski """
    wp_news = self.press_notes["wp"]  # pobranie aktualnego zbioru notatek z WP
```

```
# właściwa część metody 'scrape_wp'; iteracja po kolejnych ustalonych zakładkach
for cat in self.wp_labels:
    # zastosowanie odpowiednich metod wewnętrznych w zależności od zakładki
    if cat in ["pilka-nozna", "zimowe", "tenis", "moto", "kolarstwo"]:
        scrape_wp_sport(cat)
    else:
        scrape_wp_news(cat)
# sortowanie notatek prasowych dla WP od najnowszej do najstarzej
self.press_notes["wp"] = self.sort_notes_by_date(wp_news)
```

Metoda wewnętrzna, pobierająca notatki prasowe z *wiadomosci.wp.pl*:

```
def scrape_wp_news(category):
    """ Metoda wewnętrzna, pobierająca dane o artykułach ze strony 'wiadomosci.wp.pl'; artykuły na tamtejszych
    zakładkach są ułożone jak 'kafelki'; category - aktualnie badana kategoria """
    # pobranie całej zawartości html ze strony
    url = "https://wiadomosci.wp.pl/" + category
    soup = self._get_soup(url)
    # wyodrębnienie elementów, zawierających odnośniki do kolejnych artykułów
    article_anchors = soup.find_all("a", {"class": "a2PrHTUx"})
    # iteracja po notatkach (wyodrębninych elementach)
    for anchor in article_anchors:
        # wychwycenie i obsługa błędu, na wypadek gdyby format strony był niezgodny z obsługiwanym przez system
        try:
            # pobranie i uzupełnienie linku do oryginalnego artykułu
            # (w kodzie html dla wp odnośniki do niektórych artykułów nie posiadają nagłówków)
            art_url = anchor.get("href")
            if art_url[:7] != "https:/" and art_url[:6] != "http:/":
                art_url = "https://wiadomosci.wp.pl" + art_url
            # pobranie zawartości strony z artykułem na podstawie jego URL
            soup2 = self._get_soup(art_url)
            # zakładka 'koronawirus' zawiera notatki, pochodzące z różnych platform; program obsługuje 3 z nich
            # w tym właśnie WP
            # wyodrębnienie tytułu, lead'u i daty publikacji w zależności od strony źródłowej
            if "wiadomosci.wp.pl" in str(art_url):
                title = soup2.find("h1", {"class": "article--title a1xAmRvR"}).text.strip()
                # sprawdzenie, czy artykuł nie ma specyficznego formatu, charakterystycznego dla "WP magazyn"
                wp_magazine_href = soup2.find("div", {"data-st-area": "article-header"}).find("a").get("href")
                if wp_magazine_href == "https://magazyn.wp.pl":
                    lead = soup2.find("div", {"class": "article--lead a3x6fdyf"}).text.strip()
                else:
                    lead = soup2.find("div", {"class": "article--lead a1HGmjUl"}).text.strip()
                art_date = soup2.find("div", {"data-st-area": "article-header"}).find("time").get("datetime")[:16]
            elif "money.pl" in str(art_url):
                title = soup2.find("h1", {"class": "sc-dNLxif sc-jqCOkK ewwpFG"}).text.strip()
                lead = soup2.find("div", {"class": "sc-dNLxif sc-jqCOkK sc-gqPbQI iqrGbW"}).find_all("p")[1].text.strip()
                art_date = soup2.find("div", {"class": "sc-dNLxif bHedCF"}).find("time").get("datetime")[:16]
            elif "o2.pl" in str(art_url):
                title = soup2.find("h1", {"class": "sc-hZSUBg sc-cMhggX gwUZyd"}).text.strip()
                lead = soup2.find("div", {"class": "sc-hZSUBg sc-cMhggX sc-iuJeZd fxjjeh"}).find_all("p")[1].text.strip()
                art_date = soup2.find("div", {"class": "sc-hZSUBg lfrejV"}).find("time").get("datetime")[:16]
            # pominięcie iteracji i przejście do następnej notatki, jeżeli pętla natknie się na inną stronę
            # źródłową, niż ustalone (pozostałe wymagałyby wyszukiwania elementów html o innych wartościach atrybutów)
            else:
                continue
            # weryfikacja tytułu i daty, analogicznie do poprzednich metod
            title_repeats = self._check_if_title_repeats(wp_news, title, category)
            if title_repeats == "YES":
                break
            elif title_repeats == "MAYBE":
                continue
            art_date = art_date.replace("T", " ")
            if self.check_if_article_is_too_old(art_date):
                break
            # zapisanie danych o notatce prasowej i umieszczenie jej w zbiorze artykułów dla WP
            pressnote_info = {"Tytuł": title, "Opis": lead, "Data": art_date, "URL": art_url, "Zakładka": category}
            wp_news[title] = pressnote_info
        except AttributeError:
            continue
```

Metoda wewnętrzna, pobierająca notatki prasowe z *sportowefakty.wp.pl*:

```
def scrape_wp_sport(category):
    """ Metoda wewnętrzna, pobierająca dane o artykułach ze strony 'sportowefakty.wp.pl'; artykuły na tamtejszych
    zakładkach są ułożone w formie listy; category - aktualnie badana kategoria """
    # pobranie całej zawartości ze strony
    url = "https://sportowefakty.wp.pl/" + category
    soup = self._get_soup(url)
    # wyodrębnienie 'bloku' z poszukiwanymi artykułami; w zakładce 'kolarstwo' ma on inną nazwę w html-u
    if category == "kolarstwo":
        articles_section = soup.find("div", {"class": "column grid-wx2"})
    else:
        articles_section = soup.find("div", {"class": "column grid-wx0a1"})
    # wyodrębnienie poszczególnych 'bloczków' z notatkami prasowymi
    articles = articles_section.find_all("li", {"class": "streamshort streamshort--news"})
```

```
    # iteracja po kolejnych notatkach
    for note in articles:
        # wychwycenie i obsługa błędu, na wypadek gdyby format strony był niezgodny z obsługiwanym przez system
        try:
            # wyodrębnienie linku do oryginalnego artykułu z 'bloczka' i uzupełnienie nagłówka
            art_url = note.find("a").get("href")
            if art_url[:7] != "https:/":
                art_url = "https://sportowefakty.wp.pl" + art_url
            # pobranie zawartości strony notatki prasowej na podstawie jej URL
            soup2 = self._get_soup(art_url)
            # pobranie informacji o tytule, lead'ie i dacie publikacji oraz weryfikacja
            # (analogicznie do poprzednich metod)
            title = soup2.find("h1", {"class": "title"}).text.strip()
            title_repeats = self._check_if_title_repeats(wp_news, title, category)
            if title_repeats == "YES":
                break
            elif title_repeats == "MAYBE":
                continue
            lead = soup2.find("span", {"class": "h5"}).text.strip()
            art_date = soup2.find("address", {"class": "indicator"}).find("time", {"class": "indicator__time"}).get("datetime")[:16]
            if self.check_if_article_is_too_old(art_date):
                break
            # zapisanie danych o notatce i umieszczenie jej w zbiorze notatek prasowych dla WP
            pressnote_info = {"Tytuł": title, "Opis": lead, "Data": art_date, "URL": art_url, "Zakladka": category}
            wp_news[title] = pressnote_info
        except AttributeError:
            continue
```

Pobieranie “grupowe” (ze wszystkich platform):

```
def get_more_press_notes(self):
    """ Metoda 'główna', pobierająca notatki ze wszystkich trzech platform i wyświetlająca liczbę wszystkich
    aktualnych notatek w 'bazie' """
    self.scrape_wyborcza()
    self.scrape_onet()
    self.scrape_wp()
    self.how_many_notes_in_total()
```

Zapis pobranych danych do pliku:

```
def save_press_notes(self):
    """ Metoda zapisująca pobrane notatki prasowe do pliku press_notes.json """
    set_data_to_json("..//json_files//press_notes.json", self.press_notes)
```

Metody poboczne:

Sortowanie notatek wg daty publikacji:

```
def sort_notes_by_date(self, notes_to_sort):
    """ Sortowanie notatek prasowych z konkretnego źródła (notes_to_sort) od najnowszej do najstarszej """
    # sprawdzenie, czy zbiór notatek nie jest przypadkiem pusty
    if len(notes_to_sort) != 0:
        # iteracja po kolejnych artykułach w zbiorze
        for note in notes_to_sort:
            # zmiana formatu daty artykułów ze 'string' na 'datetime';
            # data publikacji musi mieć format: rok-miesiąc-dzień godzina:minuty
            art_date_datetime = datetime.strptime(notes_to_sort[note]["Data"], '%Y-%m-%d %H:%M')
            notes_to_sort[note]["Data"] = art_date_datetime
        # zastosowanie 'OrderedDict', w celu posortowania (malejąco) artykułów wg ich daty publikacji
        notes_ordered = OrderedDict(sorted(notes_to_sort.items(), key=lambda x: x[1]["Data"], reverse=True))
        # zmiana formatu daty notatek z powrotem na 'string' (json nie ma formatu zamiennego dla datetime)
        for note in notes_ordered:
            # ostatnie trzy znaki są ignorowane, gdyż reprezentują one sekundy (:ss), które nie są ważne dla systemu
            notes_ordered[note]["Data"] = str(notes_ordered[note]["Data"])[:-3]
        # zwrócenie posortowanego zbioru
        return notes_ordered
    # jeśli zbiór notatek jest pusty, nie ma co sortować i notes_to_sort jest z powrotem zwracane
    else:
        return notes_to_sort
```

Zmiana formatu daty z *dzień-miesiąc-rok* na *rok-miesiąc-dzień*:

```
def _modify_datetime_format(self, datetime_str):
    """ Prywatna metoda, przekształcająca format daty (łańcuch znaków) z 'dzień-miesiąc-rok czas'
    (maska: 00-00-0000 00:00) na 'rok-miesiąc-dzień czas' (maska: 0000-00-00 00:00)"""
    day = datetime_str[0:2]
    month = datetime_str[3:5]
    year = datetime_str[6:10]
    time = datetime_str[11:]
    return year + "-" + month + "-" + day + " " + time
```

Pobranie zawartości wybranej strony internetowej:

```
def _get_soup(self, url):
    """ Pobranie zawartości strony na podstawie url i przekształcenie jej na czytelny kod html """
    r = requests.get(url)  # pobranie
    soup = BeautifulSoup(r.content, features="html.parser")  # przekształcenie na odpowiedni format
    return soup
```

Weryfikacja, czy artykuł nie jest “za stary”:

```
def check_if_article_is_too_old(self, datetime_string):
    """ Kontrola daty publikacji artykułów (czy nie jest 'za wczesna') """
    # zmiana formatu daty publikacji
    datetime_obj = datetime.strptime(datetime_string, '%Y-%m-%d %H:%M')
    # porównanie z ustaloną, 'minimalną' datą; jeśli aktualny artykuł jest 'wcześniejszy', dlasze funkcje podejmą
    # odpowednie kroki
    if datetime_obj < self.min_date:
        return True
    return False
```

Weryfikacja, czy artykuł nie był już kiedyś pobierany:

```
def _check_if_title_repeats(self, press_source, title, category):
    """ Weryfikacja 'powtórek' tytułów artykułów; press_source - zbiór notatek do weryfikacji (z konkretnego portalu),
    title - tytuł aktualnego artykułu, category - aktualna zakładka badanego portalu; 'YES' - artykuł się powtarza,
    'MAYBE' - tytuł się powtarza, ale pochodzi z innej zakładki, 'NO' - artykuł jest zupełnie nowy """
    # jeśli jakimś zrządzeniem losu tytuł będzie pusty, dla zachowania bezpieczeństwa, program zwróci MAYBE''
    if title is None:
        return "MAYBE"
    # czy artykuł występuje w aktualnym zbiorze?
    if title in press_source:
        current_art = press_source[title]
        # czy pochodzi z tej samej zakładki?
        if current_art["Zakladka"] == category:
            return "YES"
        return "MAYBE"
    return "NO"
```

Wypisanie liczby pobranych notatek dla każdego z portali prasowych:

```
def how_many_notes_in_total(self):
    """ Wypisanie liczby pobranych notatek prasowych, znajdujących się w bazie """
    print("Liczba notatek prasowych w bazie:")
    total = 0  # przygotowanie licznika wszystkich artykułów
    # wypisywanie liczby notatek z poszczególnych portali
    for source in self.press_notes:
        print(f"{source.capitalize()}: {len(self.press_notes[source])}")
        total += len(self.press_notes[source])  # zwiększenie licznika 'total'
    # ile jest wszystkich artykułów razem?
    print(f"Razem: {total}")
```

PressNotesDeleter.py:

Importowane pakiety:

```
# Jan Proniewicz, 297989, Informatyka - Data Science
""" Funkcja systemu, której zadaniem jest usuwanie 'starych' notatek prasowych z obu plików .json """
from datetime import datetime  # import niezbędnej metody z modułu 'datetime'
from program_files.JsonOperations import *  # import metod, operujących na plikach .json (metody własne)
```

Inicjalizacja obiektu klasy:

```
class PressNotesDeleter:
    """ Klasa 'PressNotesDeleter', odpowiadająca za usuwanie starych notatek prasowych (proces nieodwracalny) """
    def __init__(self, max_year, max_month, max_day):
        """ Tworzenie instancji obiektu klasy 'PressNotesDeleter'; argumentami instancji są rok, miesiąc i dzień, tj.
        data, do której będą usuwane artykuły (wcześniejsze, niż ta data) """
        # uzupełnianie wartości dla miesiąca i dnia, jeśli użytkownik podał pojedyncze cyfry
        if len(str(max_month)) == 1:
            max_month = "0" + str(max_month)
        if len(str(max_day)) == 1:
            max_day = "0" + str(max_day)
        # utworzenie łańcuchowego formatu daty
        max_date_string = str(max_year) + "/" + str(max_month) + "/" + str(max_day)
        # zastosowanie metody 'datetime' w celu przekształcenia daty na domyślny format datetime
        # (rok-miesiąc-dzień godzina-minuty-sekundy); jeśli argumenty nie tworzą odpowiedniego formatu daty zgodnego z
        # wybraną maską (%Y/%m/%d), program wyrzuca wyjątek, obsługiwany w głównym menu
        self.max_date = datetime.strptime(max_date_string, '%Y/%m/%d')
        # pobranie zawartości obu plików .json
        self.press_notes = get_data_from_json("..//json_files//press_notes.json")
        self.notes_classified = get_data_from_json("..//json_files//notes_classified.json")
```

Metody główne:

Usuwanie “starych” notatek:

```
def delete_old_press_notes(self):
    """ Metoda usuwająca stare notatki prasowe z obu plików .json """
    def delete_notes_from_file(notes_data):
        """ Metoda wewnętrzna, usuwająca notatki prasowe z konkretnego pliku """
        deleted_num = 0  # licznik usuniętych notatek
        # iteracja po 'kluczach głównych', tj. portalach dla 'press_notes.json' lub kategoriach dla 'notes_classified.json'
        for main_key in notes_data:
            # wyodrębnienie kluczy notatek prasowych (ich tytułów) i odwrócenie ich kolejności;
            press_notes = notes_data[main_key]
            pn_titles = list(press_notes.keys())
            pn_titles.reverse()
            # pętla, iterująca po tytułach notatek prasowych w kolejności od najstarszej do najnowszej
            for title in pn_titles:
                current_pn = press_notes[title]
                # weryfikacja daty notatki prasowej
                datetime_obj = datetime.strptime(current_pn["Data"], '%Y-%m-%d %H:%M')
                # jeśli jest ona 'wcześniejsza' niż ustalona data 'maksymalna', notatka jest usuwana
                if datetime_obj < self.max_date:
                    press_notes.pop(title)
                    deleted_num += 1  # zwiększenie wartości licznika
                else:
                    # w przeciwnym wypadku, pętla jest przerywana, a proces - zakończony
                    break
        # zwrócenie licznika usuniętych notatek
        return deleted_num

    # część właściwa metody; zastosowanie metody wewnętrznej do usunięcia notatek z obu plików .json
    num_of_deleted = delete_notes_from_file(self.press_notes)
    delete_notes_from_file(self.notes_classified)
    # ile usunięto?
    print(f"Usunięto: {num_of_deleted}")
```

Zapisanie zmodyfikowanych zbiorów notatek do odpowiednich plików:

```
def save_modified_press_notes(self):
    """ Metoda zapisująca zmodyfikowane notatki prasove do odpowiednich plików .json """
    set_data_to_json("..//json_files//press_notes.json", self.press_notes)
    set_data_to_json("..//json_files//notes_classified.json", self.notes_classified)
```

PressNotesClassifier.py:

Importowane pakiety:

```
# Jan Proniewicz, 297989, Informatyka - Data Science
""" Funkcja systemu, odpowiedzialna za klasyfikację notatek prasowych, znajdujących się w pliku 'press_notes.json' """
from collections import OrderedDict  # import 'uporządkowanego słownika'
from datetime import datetime  # import niezbędnej metody z modułu 'datetime'
import re  # biblioteka 'regex', która przyda się podczas klasyfikacji wg liczby wystąpień słów kluczowych
from program_files.JsonOperations import *  # import metod, operujących na plikach .json (metody własne)
```

Inicjalizacja obiektu klasy:

```
class PressNotesClassifier:
    """ Klasa 'PressNotesClassifier', odpowiadająca za klasyfikację notatek prasovych """
    def __init__(self):
        """ Tworzenie instancji obiektu klasy 'PressNotesClassifier' """
        # pobranie zawartości obu plików .json
        self.notes_to_classify = get_data_from_json("..//json_files//press_notes.json")
        self.notes_classified = get_data_from_json("..//json_files//notes_classified.json")
        # sprecyzowanie słów kluczowych (niezmiennych fragmentów form) dla każdej z kategorii, oprócz 'inne';
        # w nawiasach kwadratowych znajdują się litery 'zamienne', np. 'szczepi[ćł]' oznacza 'szczepić' lub 'szczepił'
        # (technika kompatybilna z 'regex')
        politics_key_words = ["Dud[ąęy]", "Trump", "Biden", "Kaczyńsk", "Clinton", "Morawieck", "Ziobr", "Korwin", "Obam",
                              "Putin", "Łukaszenk", "Hołowni", "Trzaskowsk", "minister", "ministr", "PiS", "PO", "Sprawiedliwoś",
                              "Platform", "Konfederacj", "PSL", "trybunał", "sejm", "senat", "prezyden[ct]", "ustaw[aeiy]",
                              "opozycj", "polity[ck]", " rząd", "Rząd", "Kim Dzong", "posł", "posł[aeoó]", "konstytucj",
                              "parlament", "referendum", "Tusk", "Uni[aąei] Europejsk", "UE", "parti", "starost", "ojczyzn",
                              "nar[oó]d", "Rad[ayz]", "premier", "Gowin", "prawic", "lewic", "koalicj", "demokrac", "Kapitol",
                              "autorytar", "totalitar", "wojn[aąeiy]", "wojenn", "komisj", "przywódc", "protest", "Paszynian",
                              "Alijew", "wybor[aąóy]", "wyborcz", "ONZ", "NATO", "Kongres", "Kosiniak", "ambasador", "Macierewicz",
                              "smoleńsk", "spisek", "spisk", "generał", "sekretarz", "demonstracj", "Nawaln", "zwolenni[ck]"]
        football_key_words = ["piłkarz", " bramk", "Bramk", "Lewandowsk", "Messi", "Cristiano Ronaldo", "mundial", "nożn[ąae]"]
        wintersport_key_words = ["narty", "nartach", "narciarski", "Stoch", "Kuback", "Ży[łł][ąęy]", "Stęka[łł]",
                                 "skoczni", "łyżwiar", "łyżw[ay]", "hokej", "curling", "lodowisk"]
        tenis_key_words = [" tenis", "Tenis", "Radwańsk", "Hurkacz", "Australian Open"]
        moto_key_words = ["Formu", "F1", " bolid", "Bolid" "Kubic", "Ferrari", "Schumacher", "rajdow[iy]"]
        cycling_key_words = [" rower", "Rower", " kolarz", "Kolarz", "Tour de", "peleton", "kolarsk"]
        sports_key_words = ["sport", "zawod[aóy]", "mistrz", "zawodnik", "drużyn", "turniej", "medal", "kibic", "doping",
                            "igrzysk", "olimpiad", "olimpijsk", "trener"]
        coronavirus_key_words = ["koronawirus", "pandemi", "COVID-19", "SARS-CoV-2", "obostrze", "kwarantann", "lockdown",
                                 "szczepi[eo][nń]", "szczepi[ćł]", "restrykcj[ei] rząd", "restrykcjach rząd",
                                 "restrykcjom rząd", "epidemi", "sanitarn", "covid"]
        health_key_words = ["szpital", "medyc", "WHO", "NFZ" "Zdrowia", "lekarstw", "chirurg", "przeszczep", "lekarz",
                            "okulist", "przychodni", "chor[oó]b", "medyk", "pogotowi", "ratowni[ck]", "pacjent",
                            "nowotw[oó]r", "pielęgniar", "diagnoz", "TOPR", "szkod[lz]"] + coronavirus_key_words
        culture_key_words = ["film", "utw[oó]r", "piosenk", "książk", "autor", "serial", "zabytk", "zabytek", "teatr",
                             "kin[ao]", "kinie", "muzyk", "muzyczn", "koncert", "koncercie", "sztuk", "twórc",
                             "arty[sś][ct]", "Mickiewicz", "romanty[cz]", "renesans", "literatur", "literack", "ballad",
                             "kultur", "muzeum", "histori", " śpiew", "Śpiew", "przeb[oó]j"]
        sciandtech_key_words = ["kosmos", "NASA", "kosmiczn", "galakty", "odkryci", "odkrywać", "naukow[ci]", "archeolo",
                                "paleontolo", "geolog", "wynalaz", "wynalez", "YouTube", "Spotify", "Twitter", "Facebook",
                                "CD Project", "komputer", "epok[ai]", "epokow", "histori", "patent", "DNA", "specjalist",
                                "bada[jąn]", "innowac", "uniwersyte", "uczelni", "akademi", "przełom", "nauk[ai]", "technologi",
                                "rozw[oó]j", "zdoby[ct]", "serwis", "użytkowni", "portal", "anali[zt]", "medi[aó]"]
        economy_key_words = ["przedsiębiorstw", "przedsiębiorc[aóy]", "inflacj", "doch[oó]d", "podatk", "bank", "restaura[ct]",
                             "biznes", "koszt", "splajtow", "usług", "finans", "strat", "gospodar", "turyst", "turyści",
                             "tys. zł", "mln zł", "mld zł", "tysiąc zł", "tysiące zł", "tysięcy zł" "milion zł", "miliony zł",
                             "milionów zł", "miliard zł", "miliardy zł", "miliardów zł", "dolarów", "funtów", "euro",
                             "pieni[ąę]dz", "Skarb", "bitcoin", "500 plus", "wypłac[ao]n", "podwyżk", "obniżk", "obniżek",
                             "podwyżek" "budże[ct]", "knajp", "branż", "wynagrodze[nń]", " cię[cć]", "Cię[cć]", "opłat",
                             "przetarg"]
```

```
        criminal_key_words = ["zaginion", "zagin[ąę]ł", "mordowan", "morderstw", "morderc", "zabójstw", "zabójc", "kradzież",
                              "kradzion", "porwan", "narkotyk", "przemyt", "przemycenie", "przmycon", "terroryst", "policj",
                              "milicj", "mandat", "wypadek", "wypadk", "straż", "strajk", "zamieszki", "zamieszkach",
                              "zamieszek", "Kapitol", "plądrowa", "zaatakowa", "podejrzan", "sprawc[aoóy]", "skazan", "wyrok",
                              "wiezi[eo]n", "pozbawieni[ae] wolności", "dożywoci[ae]", "legaln", "promil", "nietrzeźw",
                              "zwłok", "śledztw", "przestępstw", "prokurat[ou]r", "sąd", "imitacj", "podróbk", "podr[ao]bion",
                              "pożar", "spion[ąę]ł", "uratowa", "zatru[cć]ł", "otru[cć]ł", "alarmow", "pod wpływem alkoholu",
                              "potrąci", "potrącon", "zatrzyma[ćn]", "poszukiwan", "tragedi", "tragiczn", "nie ży[jł]",
                              "śmier[cć]", "śmiertelн", "życi", "żyć", "przestępcz", "fałszerstw", "fałszowan", "wyłudz[aeln]ie",
                              "nieprzytomn", "rann[aiy]", "zderzy", "zderz[eo]n", "złodzie[ij]", "zarzut", "zabił", "zabici",
                              "przeży[ćł]", "strzel[ai]", "n[oó]ż", "pomoc", "zbrodni", "ciał[ao]", "zgin[ąę][łl]", "rozbi[łt]",
                              "martw[aiy]", "apelacj", "FBI", "uzbroj[eo]n", "sprawiedliwoś", "śledcz", "aresz[ct]", "CIA",
                              "samobój[cs]", "katastrof", "wyciek", "ewaku[ao]", "pijan", "sędzi", "włam[ay][łlnw]",
                              "napastni[ck]", "uderzy[ćł]", "zmarł", "gwał[ct]", "molestowan", "nieletn"]
        religion_key_words = ["kości[eoó][łl]", "duchown", "chrześcija", "ksiądz", "księża", "księży", "kapłan", "biskup",
                              "zakon", "papież", "papieski", "islam", "muzułman", "prawosław[in]", "katolic", "katolizm",
                              "żydowsk", "Żyd", "chrzest", "chrzczon", "bierzmowan", "świet[ay]", "Boże", "Wigili", "świątecz",
                              "najświętsz", "koled", "Wielkanoc", "Jezus", "Chrystus", "B[oó]g", "msz[ay]", "modli[ćł]",
                              "kaplic", "diecezj", "katedr", "misjonar", "wiern[aiy]", "religi", "wierze[nń]", "dominikan",
                              "franciszkan", "benedyktyn", "Mary[ij]", "ojc[aeiu]", "proboszcz", "parafi"]
        # umieszczenie zbiorów słów kluczowych do jednej listy
        key_words_list = [politics_key_words, football_key_words, wintersport_key_words, tenis_key_words, moto_key_words,
                          cycling_key_words, sports_key_words, health_key_words, coronavirus_key_words, culture_key_words,
                          sciandtech_key_words, economy_key_words, criminal_key_words, religion_key_words]
        # wylistowanie możliwych kategorii, poza 'inne'
        self.categories_list = ["polityka", "piłka nożna", "zimowy sport", "tenis", "sport motorowy", "kolarstwo",
                                "sport", "medycyna", "koronawirus", "kultura", "nauka i technika", "gospodarka",
                                "kryminalne i wypadki", "religia i Kościół"]
        # uporządkowanie słów kluczowych w słowniku: klucz=kategoria, wartość=słowa kluczowe
        self.key_words_for_categories = {}
        for i in range(len(self.categories_list)):
            self.key_words_for_categories[self.categories_list[i]] = key_words_list[i]
```

Metody główne:

Kategoryzacja notatek prasowych (z podziałem na dwa ustalone sposoby):

Kod właściwy:

```
def classify_press_notes(self):
    """ Metoda klasyfikująca notatki prasowe na dwa sposoby. """
```

```
# właściwa część metody 'classify_press_notes'
# iteracja po kolejnych zbiorach notatek z konkretnych portali
for notes_source in self.notes_to_classify:
    press_notes = self.notes_to_classify[notes_source]
    # iteracja po kolejnych notatkach (tytułach)
    for title in press_notes:
        # weryfikacja, czy dana notatka już została gdzieś zaklasyfikowana
        if self.check_if_already_classified(title):
            # jeśli tak, petla zostaje przerwana (można przypuszczać, że nastepne notatki też są zaklasyfikowane)
            break
        # przygotowanie listy potencjalnych kategorii dla notatki prasowej
        categories_for_note = []
        # wyszukiwanie potencjalnych kategorii dla aktualnej notatki prasowej
        # (najpierw wg zakładki, potem - po słowach kluczowych)
        the_note = press_notes[title]
        classify_by_label(the_note)
        classify_by_key_words(the_note)
        # zapisanie notatki prasowej w zbiorach, odpowiadajacych zebranym kategoriom
        for cat in categories_for_note:
            notes_from_cat = self.notes_classified[cat]
            notes_from_cat[title] = the_note
        # jeśli dla notatki nie znaleziono żadnej kategorii, zostaje ona przydzielona do zbiotu 'inne'
        if len(categories_for_note) == 0:
            self.notes_classified["inne"][title] = the_note
# posortowanie notatek w każdej kategorii w kolejności od najnowszej do najstarszej
self.sort_notes_by_date()
# wypisanie, ile notatek znajduje się w każdej z kategorii
self.how_many_notes_in_categories()
```

Metoda wewnętrzna, kategoryzująca notatki wg ich zakładek:

```
def classify_by_label(note_to_classify):
    """ Metoda wewnętrzna klasyfikująca notatki prasowe wg zakładek, z których pochodzą; note_to_classify -
    - aktualna notatka """
    label = note_to_classify["Zakladka"]  # wyodrębnienie zakładki notatki prasowej
    # zbadanie zakładki i specyfikacja kategorii, która by jej odpowiadała
    if label == "polityka":
        category = self.categories_list[0]
    elif label in ["sport", "pilka-nozna", "skoki-narciarskie", "zimowe", "formula-1", "moto", "tenis",
                   "kolarstwo"]:
        category = self.categories_list[6]
        # notatki z zakładek, odpowiadającym konkretnym dyscyplinom, będą należeć zarówno do kategorii sport,
        # jak i kategorii dla ich deyscypliny
        category2 = ""
        if label == "pilka-nozna":
            category2 = self.categories_list[1]
        elif label in ["skoki-narciarskie", "zimowe"]:
            category2 = self.categories_list[2]
        elif label in ["formula-1", "moto"]:
            category2 = self.categories_list[4]
        elif label == "tenis":
            category2 = self.categories_list[3]
        elif label == "kolarstwo":
            category2 = self.categories_list[5]
        if category2 != "":
            categories_for_note.append(category2)
    elif label in ["zdrowie", "koronawirus"]:
        category = self.categories_list[7]
        # z reguły, notatki z zakładki 'koronawirus' są mocno powiązane z 'medycyną'
        if label == "koronawirus":
            categories_for_note.append(self.categories_list[8])
    elif label == "kultura":
        category = self.categories_list[9]
    elif label in ["nauka", "technika"]:
        category = self.categories_list[10]
    elif label == "gospodarka":
        category = self.categories_list[11]
    elif label == "religia":
        category = self.categories_list[13]
    # jeśli zakładka nie odpowiada żadnej kategorii, dalsze operacje są pomijane
    else:
        category = None
    if category is not None:
        # umieszczenie zidentyfikowanej kategorii na liście potencjalnych kategorii dla notatki
        # (zaimplementowana w metodzie nadrzędnej)
        categories_for_note.append(category)
```

Metoda wewnętrzna, kategoryzująca notatki wg liczby wystąpień “wyrażeń klucz”:

```
def classify_by_key_words(note_to_classify):
    """ Metoda wewnętrzna klasyfikująca notatki prasowe na podstawie wystąpień określonych słów kluczowych;
     note_to_classify - aktualna notatka"""
    # klasyfikacja bazuje na liczbie wystąpień określonych wyrażeń kluczowych w tytule i lead'ie (opisie)
    note_title = note_to_classify["Tytuł"]
    note_lead = note_to_classify["Opis"]
    text_to_check = note_title + " " + note_lead

    def classify_by_single_occurence(current_category, keyword):
        """ Kolejna metoda wewnętrzna, która klasyfikuje notatkę na podstawie pojedynczego wystąpienia
        słowa kluczowego (keyword); current_category - aktualnie badana kategoria """
        # metoda 'search' biblioteki 're(gex)' wychwytuje pierwsze wystąpienie wyrażenia w 'stringu'
        if re.search(keyword, text_to_check) is not None:
            # jeśli wyrażenie zostanie znalezione, kategoria jest dodawana do listy potencjalnych
            categories_for_note.append(current_category)
            # dodatkowo, jeśli notatka należała do jednej z dyscyplin sportu, to zostaje jej również
            # przydzielona kategoria 'sport' (tego samego nie robi dla 'koronawirusa' i 'medycyny', gdyż
            # nie każde wydarzenie związane z koronawirusem jednoznacznie wiąże się z medycyną)
            if current_category in ["piłka nożna", "zimowy sport", "tenis", "sport motorowy", "kolarstwo"]:
                categories_for_note.append("sport")
```

```
            # zwracane wartości wpływają na dalsze prcesy w metodzie nadrzędnej
            return True
        else:
            return False

    # iteracja po kolejnych kategoriach
    for category in self.key_words_for_categories:
        # dlasze procedury są pomijane, jeżeli kategoria już się znajduje na liście
        if category not in categories_for_note:
            key_words = self.key_words_for_categories[category]  # pobranie słów klucz
            num_of_occurrences = 0  # licznik wystąpień słów kluczowych w tytule i opisie notatki
            # iteracja po kolejnych słowach kluczowych
            for k_word in key_words:
                # jeśli kategoria reprezentuje jedną z określonych dyscyplin sportu lub koronawirusa,
                # dokonuje się klasyfikacji na podstawie pojedynczego wystąpienia (ponieważ są to tematyki o
                # bardziej zawężonym zakresie; wciąż wiążą się z tym błędy klasyfikacji)
                if category in ["piłka nożna", "zimowy sport", "tenis", "sport motorowy", "kolarstwo",
                                "koronawirus"]:
                    # jeśli się uda zaklasyfikować notatkę, petla jest przerywana i przechodzi do kolejnej kategorii
                    if classify_by_single_occurence(category, k_word):
                        break
                    # w przeciwnym wypadku, ponawia proces dla tego samego słowa, zaczynającego się od dużej litery
                    else:
                        if k_word.islower():
                            k_word_capitalized = k_word.capitalize()
                            if classify_by_single_occurence(category, k_word_capitalized):
                                break
                # standardowo, klasyfikacja odbywa się poprzez zliczanie wystąpień wyrażeń kluczowych
                else:
                    # metoda 'findall' biblioteki 're(gex)' generuje listę wystąpień badanego wyrażenia w 'stringu'
                    occurrences_list = re.findall(k_word, text_to_check)
                    # zwiększenie licznika wystąpień o długość listy wystąpień
                    num_of_occurrences += len(occurrences_list)
                    # powtórzenie tej samej procedury dla tego samego wyrażenia, zaczynającego się od dużej litery
                    if k_word.islower():
                        k_word_capitalized = k_word.capitalize()
```

Zapis posegregowanych notatek prasowych do pliku:

```
def save_classification_changes(self):
    """ Zapisanie zaklasyfikowanych notatek do pliku notes_classified.json """
    set_data_to_json("..//json_files//notes_classified.json", self.notes_classified)
```

Metody poboczne:

Sortowanie notatek wg daty publikacji:

```
def sort_notes_by_date(self):
    """ Sortowanie notatek prasowych w poszczególnych kategoriach od najnowszej do najstarszej """
    # iteracja po kategoriach
    for cat in self.notes_classified:
        notes_to_sort = self.notes_classified[cat]
        # jeśli zbiór dla danej kategorii jest pusty, dalsze kroki są pomijane
        if len(notes_to_sort) != 0:
            # zastosowanie zaimportowanej metody 'datetime', w celu zmiany formatu daty ze 'string' na datetime
            # w każdej z notatek prasowych
            for note in notes_to_sort:
                art_date_datetime = datetime.strptime(notes_to_sort[note]["Data"], '%Y-%m-%d %H:%M')
                notes_to_sort[note]["Data"] = art_date_datetime
            # wykorzystanie 'uporządkowanego słownika', w celu posortowania (malejąco) notatek prasowych wg daty publikacji
            notes_ordered = OrderedDict(sorted(notes_to_sort.items(), key=lambda x: x[1]["Data"], reverse=True))
            # zmiana formatu daty z powrotem na 'string' (json nie ma formatu zamiennego dla datetime)
            for note in notes_ordered:
                # ostatnie trzy znaki w 'datetime' reprezentują sekundy, które są niepotrzebne
                notes_ordered[note]["Data"] = str(notes_ordered[note]["Data"])[:-3]
            # wprowadzenie uporządkowanego zbioru do całego zbioru sklasyfikowanych notatek
            self.notes_classified[cat] = notes_ordered
```

Resetowanie przydziału artykułów do kategorii:

```
def restart_classification(self):
    """ Reset, tj. opróżnianie zbioru sklasyfikowanych notatek prasowych, żeby było można je podzielić od nowa """
    for category in self.notes_classified:
        self.notes_classified[category] = {}
```

Weryfikacja, czy artykuł już został gdzieś zaklasyfikowany:

```
def check_if_already_classified(self, title):
    """ Metoda dprawdzająca, czy dana notatka (title) jest już gdzieś zaklasyfikowana """
    # iteracja po poszczególnych kategoriach w pliku notes_classified.json
    for category in self.notes_classified:
        notes_from_cat = self.notes_classified[category]
        if title in notes_from_cat:
            return True
    return False
```

Wypisanie liczby notatek prasowych w każdej z kategorii:

```
def how_many_notes_in_categories(self):
    """ Zliczanie notatek prasowych w każdej z kategorii """
    print("Liczba artykułów/notatek w każdej z kategorii:")
    for category in self.notes_classified:
        print(f"{category.capitalize()}: {len(self.notes_classified[category])}")
```

PressNotesVisualizer.py:

Importowane pakiety:

```
# Jan Proniewicz, 297989, Informatyka - Data Science
""" Funkcja systemu, odpowiadająca za 'wizualizację' (po prostu - wypisywanie) artykułów z konkretnych kategorii """
from program_files.JsonOperations import *  # import metod, operujących na plikach .json (metody własne)
```

Inicjalizacja obiektu klasy:

```
class PressNotesVisualizer:
    """ Klasa 'PressNotesVisualizer', odpowiadająca za 'prezentację' sklasyfikowanych notatek prasowych """
    def __init__(self):
        """ Utworzenie instancji obiektu klasy """
        # pobranie zawartości pliku 'notes_classified.json'
        self.notes_classified = get_data_from_json("..//json_files//notes_classified.json")
        # sprecyzowanie listy kategorii
        self.categories_list = list(self.notes_classified.keys())
```

Metody główne:

Wypisanie notatek z konkretnej kategorii:

```
def show_press_notes_from_category(self, cat_index):
    """ Metoda wypisująca notatki prasowe z jednej kategorii; cat_index - indeks kategorii w stosunku do ustalonej
    listy kategorii """
    # pobranie zbioru notatek dla wybranej kategorii
    category = self.categories_list[int(cat_index)]
    notes_from_cat = self.notes_classified[category]
    # jeśli zbiór nie jest pusty...
    if len(notes_from_cat) != 0:
        # wyodrębnienie kluczy słownika (tj. tytułów artykułów) i odwrócenie ich kolejności;
        # będą one wypisywane w kolejności od najstarszej do najnowszej, ponieważ na konsoli
        # wygodniej będzie czytać artykuły od dołu do góry
        pn_titles = list(notes_from_cat.keys())
        pn_titles.reverse()
        # iteracja po kolejnych tytułach/notatkach i wypisanie wszystkich informacji w ustalonym formacie
        for title in pn_titles:
            note = notes_from_cat[title]
            print(f"Tytuł: \"{note['Tytuł']}\"")
            print(note["Opis"])
            print(f"Data: {note['Data']}")
            print(f"Link do artykułu: {note['URL']}")
            print()
        # napisanie, ile notatek znajduje się łącznie w danej kategorii
        print("Razem: " + str(len(notes_from_cat)))
    # wyświetlenie odpowiedniego komunikatu, jeżeli zbiór notatek z danej kategorii jest pusty
    else:
        print("Kategoria pusta!")
```

Wywołanie menu kategorii notatek prasowych:

```
def continuous_visualization(self):
    """ Metoda otwierająca osobne menu dla 'wyświetlacza' notatek prasowych """
    # 'włączenie' menu, działającego na zasadzie pętli
    visualizing_enabled = True
    while visualizing_enabled:
        print("Notatki z której kategorii chciałbyś zobaczyć?")
        # wybór kategorii (podanie jednej cyfry z poniższych lub 'q'), z której notatki mają być wypisane;
        # jeśli użytkownik poda indeks wychodzący poza zakres listy kategorii lub wprowadzi dane innego typu,
        # program wraca do menu głównego na drodze obsługi wyjątku (w głównym menu)
        cat_index = input("0 - polityka;\n1 - sport;\n2 - piłka nożna;\n3 - zimowy sport;\n4 - tenis;\n5 - sport "
                          "motorowy;\n6 - kolarstwo;\n7 - medycyna;\n8 - koronawirus;\n9 - kultura;\n10 - nauka i "
                          "technika;\n11 - gospodarka;\n12 - kryminalne i wypadki;\n13 - religia i Kościół;\n"
                          "14 - inne.\nNaciśnij 'q' aby wyjść.\n")
        # przerwanie pętli, jeśli użytkownik zdecydował się wyjść
        if cat_index == 'q':
            visualizing_enabled = False
        # zastosowanie metody, wypisującej wszystkie notatki dla wybranej kategorii (dokładniej mówiąc: jej indeksu)
        else:
            self.show_press_notes_from_category(cat_index)
        print()
```