Politechnika Łódzka

Wydział Elektrotechniki, Elektroniki, Informatyki i Automatyki

Instytut Informatyki Stosowanej

**PRACA DYPLOMOWA MAGISTERSKA**

Ekstrakcja informacji z utworu literackiego

Extracting essential data from a literary work

Paweł Wolski

Nr albumu: 234014

Dr hab. Szymon Grabowski, prof. PŁ

Łódź, wrzesień 2021

*Składam serdeczne podziękowania dr hab. Szymonowi Grabowskiemu za poświęcony czas i pomoc w przygotowaniu niniejszej pracy*

Streszczenie:

Praca dyplomowa proponuje rozwiązanie problemu analizy długich tekstów, m.in. książek za pomocą automatycznych narzędzi dostarczanych przez biblioteki zawarte w języku skryptowym Python. Większość rozwiązań NLP jest przeznaczona dla aplikacji mobilnych lub rozwiązań sieciowych. Praca opisuje historię przetwarzania języka naturalnego od jego powstania, jak i pierwszej implementacji aplikacji komputerowej, aż po dzisiejsze sposoby na wykorzystywanie go. Przygotowano specjalny program pomagający w automatycznej ekstrakcji informacji z dzieła literackiego. Aplikacja została napisana w języku Python 3.7.9 wraz z bibliotekami spaCy, pyqt oraz pandas. Program posiada interfejs użytkownika oraz instrukcję instalacji wraz z opisem poruszania się po nim. Aplikacja została również przetestowana za pomocą testów jednostkowych wraz z statystykami pokrycia kodu testem, jak i testami użytkownika.

Słowa kluczowe: tokenizacja, dzieło literackie, języki skryptowe, Python, NLP

Summary:

The diploma thesis presents a way to solve the problem of analyzing long texts, like books using automatic tools provided by libraries included in the Python scripting language. Most NLP solutions are designed for mobile applications or web solutions. The work describes the history of natural language processing from its inception as well as the first implementation of a computer application, to today's ways of using it. A special program has been prepared to assist in the automatic extraction of information from a literary work. The application was written in Python 3.7.9 with the spaCy, pyqt and pandas libraries. The program has a user interface and installation instructions with a description of how to navigate through it. The application has also been tested using unit tests along with code coverage statistics, as well as user tests.

Keywords: tokenization, literary work, scripting languages, Python, NLP

Spis treści

[1. Wstęp 8](#_Toc81418058)

[2. Cel i zakres pracy 9](#_Toc81418059)

[3. Algorytmy przetwarzające tekst 10](#_Toc81418060)

[3.1. Historia NLP 11](#_Toc81418061)

[3.2. Podstawowe operacje 14](#_Toc81418062)

[3.3. Podobne rozwiązania 18](#_Toc81418063)

[4. Opis środowiska i aplikacji 20](#_Toc81418064)

[4.1. Język Python 20](#_Toc81418065)

[4.2. Biblioteka spaCy 22](#_Toc81418066)

[4.3. Biblioteka pyqt 25](#_Toc81418067)

[5. Opis metryk 27](#_Toc81418068)

[5.1. Atrybuty długich teksów pisanych 27](#_Toc81418069)

[5.2. Metryki trudności czytanego tekstu 32](#_Toc81418070)

[5.3. Porównanie zebranych danych 34](#_Toc81418071)

[5.4. Miejsce i czas akcji 43](#_Toc81418072)

[6. Projekt aplikacji do ekstrakcji informacji z dzieła literackiego 46](#_Toc81418073)

[6.1. Założenia aplikacji 46](#_Toc81418074)

[6.2. Wymagania aplikacji 48](#_Toc81418075)

[6.3. Obsługa aplikacji 49](#_Toc81418076)

[6.4. Zarys implementacji 56](#_Toc81418077)

[6.5. Opis aplikacji 57](#_Toc81418078)

[6.6. Testy aplikacji 59](#_Toc81418079)

[7. Wnioski 62](#_Toc81418080)

[7.1. Wnioski badawcze 62](#_Toc81418081)

[7.2. Drogi rozwoju aplikacji 62](#_Toc81418082)

[Literatura 64](#_Toc81418083)

# Wstęp

Rozwój technologiczny sprawił, że w dzisiejszych czasach komputer jest potrzebny w każdej dziedzinie życia. Jest odpowiedzialny za wiele aspektów w funkcjonowaniu każdego człowieka, od pracy zawodowej, edukacji poprzez spędzanie czasu wolnego, jak i również rozwój osobisty. Każda z wymienionych dziedzin jest rozważana oraz niesie ze sobą wiele dróg rozwoju. Ciągnie to za sobą rozwój wielu platform, aby jak najbardziej ułatwić życie człowieka, stawiając przed nim coraz nowsze wyzwania.

Zgodnie z tytułem pracy, jedną z dziedzin rozwoju osobistego człowieka, jak i spędzania wolnego czasu jest czytanie książek. Rozwija to umiejętności czytania ze zrozumieniem, jak i pobudza wyobraźnię. Z biegiem czasu przybywa coraz więcej książek i autorów. W dzisiejszych czasach bardzo też popularne stało się pisanie książek przez zwykłych ludzi, dlatego liczba dzieł znacząco wzrosła. Człowiek z bardzo dużą ilością informacji do przetworzenia na co dzień nie jest w stanie przeczytać każdej książki. Ponadto szczegółowa analiza dłuższego utworu (np. powieści) zajęłaby dużo czasu i wielu ekspertów nie byłoby w stanie poradzić sobie z każdym tytułem na bieżąco.

W pracy podjęto próbę analizy i stworzenia systemu, który by automatycznie przetwarzał informacje znajdujące się w dziełach literackich, dając użytkownikowi końcowe informacje do przeanalizowania i wyciągnięcia wniosków opisujących daną książkę, ale także styl autora jaki został w niej użyty. Jej celem jest stworzenie aplikacji wyciągającej najważniejsze informacji z danego dzieła literackiego, statystycznych wartości gramatycznych, podzielonych także na lokalne części w przypadku książki, w której występują rozdziały. Końcowy użytkownik nie musiałby czytać utworu wiele razy, aby zrozumieć jego znaczenie, a mógłby już zająć się utworem mając wynik wstępnej analizy dzieła przed sobą.

# Cel i zakres pracy

Celem pracy jest stworzenie aplikacji na komputer PC, która będzie wyciągała najważniejsze informacje zawarte w dziele literackim, mogące stanowić metryki opisujące styl danego autora. Zakres pracy obejmuje:

* zaczerpnięcie informacji z literatury nt. tworzenia aplikacji w języku Python, aplikacji okienkowych oraz pokrycia testem kodu
* przegląd literatury o tematyce przetwarzania maszynowego języka naturalnego oraz dostarczonych bibliotek do języka Python
* wybór miar i algorytmów wykorzystywanych w projektach, a także opracowanie logiki aplikacji
* implementację danego rozwiązania, obiektów metod i odpowiedniego interfejsu użytkownika
* zadbanie o odpowiednią jakość kodu i pokrycie go testami
* analizę zebranych danych, mogących określić styl danych autorów, dla których były prowadzone badania.

# Algorytmy przetwarzające tekst

Wiele nowoczesnych algorytmów skupia się na przetwarzaniu tekstu. Dana struktura może mieć różne właściwości albo reprezentacje i jest wielokrotnie przetwarzana przez każdego programistę w trakcie jego pracy. Brakuje uniwersalnej definicji NLP (Natural Language Processing). Według definicji słownikowej jest to analiza tekstu przez komputer i lub generowanie go w języku naturalnym, zrozumiałym dla człowieka. Mimo iż definicja jest poprawna, nie zawsze oddaje całość obrazu algorytmów zajmujących się przetwarzaniem tekstu [[2]](#_2._[Online]_https://ai.pwn.pl/blog/).

Pierwszym aspektem jest przetwarzanie tekstu, gdy odbiorcą danych w postaci wejściowej języka naturalnego jest komputer. Drugim natomiast aspektem jest generowanie języka, o ile komputer jest w stanie prezentować dane cyfrowe w postaci komunikatu przedstawionego w języku naturalnym [[1]](#_E._D._Liddy,). Mimo iż mają one wiele wspólnego w zakresie metodologii i stosowanych technik przetwarzania, jest to bardziej adekwatna, rozwinięta i złożona domena, podczas gdy generowanie dodatkowo potrzebuje celu określania strategii generacji języka.

Można wyróżnić podstawowe domeny Natural Language Processing na podstawie podejścia zastosowanego do analizy danych: Symboliczne NLP oraz Statystyczne NLP. Pierwsze podejście opiera się na wykorzystaniu określonych algorytmów i reguł w oparciu o analizę zjawisk językowych oraz schematy reprezentacji wiedzy. Był to sposób używany we wczesnych wersjach NLP do kategoryzowania tekstu, rozwiązywania niejasności oraz ekstrakcji informacji. Drugie podejście opiera się na Statystycznym NLP, czyli wykorzystaniu technik matematycznych do trenowania modelu z dużą liczbą tekstów etykietowanych, zwanych także korpusami. Zjawiska językowe są modelowane także na podstawie rzeczywistych przykładów. Ma podejście statystyczne zastosowania w takich obszarach, jak analizowanie i rozpoznawanie mowy, oznakowanie części mowy i wyszukiwanie i tłumaczenie danego tekstu maszynowo [[2]](#_2._[Online]_https://ai.pwn.pl/blog/).

Inny podział zależy od technik, które stosuje się do analizy danych. Dane techniki są powiązane z działami językoznawstwa. Jedną z nich jest analiza morfologiczna, która w celu zbadania danych morfemów zawartych w słowach używa fragmentów o uniwersalnym znaczeniu. Drugie podejście oparte na analizie leksykalnej polega na interpretacji znaczenia słowa i części mowy. Kolejnym podejściem zajmującym się wykrywaniem struktury gramatycznej zdania oraz określeniem zależności między słowami jest analiza składniowa. Następnym przykładem jest analiza semantyczna, gdy sprawdzane słowo ma wiele znaczeń. Dokonuje się tego poprzez badanie relacji między znaczeniami słowa w zdaniu. Analiza pragmatyczna jest stosowana do wyszukiwania informacji z otaczającego kontekstu. Należy wtedy przeanalizować tekst, aby wywnioskować dodatkowe znaczenie kryjące się za nim. Czasami jest niemożliwe, aby określić, do kogo odnoszą się zaimki osobowe, bez wykorzystania specjalnej metody. Analiza fonologiczna jest używana tylko w obszarze rozpoznawania mowy NLP. Odwołuje się także do dźwiękowej natury danego języka wraz z ustaleniem cech fonologicznych, ważnych w komunikacji językowej [[5]](#_[Online]_https://monkeylearn.com/se).

Ponieważ jednym z celów NLP jest symulacja ludzkiego zachowania można je uznać za jedną z kategorii wykorzystania Sztucznej Inteligencji. We wczesnej fazie NLP było określane jako naturalne zrozumienie języka. Fakt iż obecne systemy można sparafrazować za pomocą wprowadzanego tekstu i odpowiedzi na pytanie na jego temat, może dać rezultat w postaci informacji oraz wnikliwych wniosków z niego. Na chwilę obecną ludzki poziom poznania nadal pozostaje poza zasięgiem NLP, co może być w przyszłości jego ostatecznym celem.

## Historia NLP

Idea komputerowego przetwarzania języka pojawiła się już we wczesnych epokach informatyki. W latach 50. słynny naukowiec Alan Turing, uważany za twórcę sztucznej inteligencji, próbował odpowiedzieć w swojej pracy na pytanie „Can Machines think?” [[3]](#_A._M._Turing,). Dokładnie nie udało mu się określić jaki jest akt myślenia przez maszynę, dlatego zaproponował odpowiednie rozwiązanie problemu, który mógłby być analogiczny do pierwotnego. Zasugerował, że sztuczna inteligencja byłaby w stanie oszukać ludzkiego sędziego, tj. przekonać go, iż ma do czynienia z człowiekiem. Orzeczenie musiało być wydane na podstawie nawiązanej między nimi rozmowy. Dlatego podstawowym wymaganiem jakie musiałby spełniać w takiej sytuacji komputer była zdolność do interpretowania języka w sposób naturalny [[3]](#_3._A._M.).

Pierwsze próby wykorzystania komputera do badań nad analizą języka zostały podjęte w latach 50. Następstwa drugiej wojny światowej, po której w historii nastąpiła zimna wojna, zwiększyły zapotrzebowanie na rosyjskich i niemieckich tłumaczy, którzy pomogli by w tłumaczeniu istotnych dla brytyjskich i amerykańskich naukowców dokumentów oraz rządów. Badania prowadzone w tym okresie miały na celu automatyzację pracy tłumaczy i skoncentrowanie się na tłumaczeniu maszynowym NLP. Od czasu osiągnięcia zbieżnej technologii z zainteresowaniem rządu USA, badania były w dużej mierze dotowane przez pieniądze państwowe.

Najwcześniejszym podejściem do tłumaczenia maszynowego tekstów było użycie automatycznego słownika, aby przetłumaczyć każe słowo z osobna i podporządkować je zgodnie z zasadami języka docelowego. Wielokrotne próby informatyków zakończyły się niepowodzeniem. Napotkali problemy ze złożonością leksykalną i gramatyczną oraz wieloznacznością pewnych słów. Nawet dwujęzyczni naukowcy mieli trudności z zastosowaniem swojej wiedzy w dziedzinie programowania. Pomoc lingwistów była niezbędna do pomyślnego przeprowadzenia badań. Pomimo obiecujących postępów w rozwoju językoznawstwa teoretycznego i algorytmów przetwarzania języka, naukowcom nie udało się osiągnąć zadowalających efektów, ponieważ zasady językowe okazały się znacznie bardziej złożone i skomplikowane niż przypuszczano, a ówczesne komputery nie były zbyt mocne, aby poradzić sobie z tłumaczeniem w rozsądny sposób, a czas na rozważenie przydatności takich metod był krótki. Yehoshua Bar-Hillel, pracownik MIT w dziedzinie tłumaczenia maszynowego twierdził, że w dającej się przewidzieć przyszłości, pełne zautomatyzowanie tłumaczenia w wysokiej jakości nie będzie możliwe, ponieważ komputerom nadal brakuje ludzkich umiejętności, takich jak między innymi zrozumienie kontekstu i analiza znaczenia, aby prawidłowo wybrać tłumaczenie danego słowa, jeśli ma więcej niż jedno znaczenie [[4]](#_Y._Bar-Hillel,_"Appendix).

ALPAC (Komitet Doradczy ds. Automatycznego Przetwarzania Języka) wydał w 1966 roku raport, w którym wskazywał na problemy jakościowe i szybkościowe oraz kosztowe MT. Ta opinia polegała na tym, iż po 8 latach pracy nad projektem na Uniwersytecie Georgetown wytworzone dokumenty wejściowe wymagały zaksięgowania [[7]](#_[Online]_https://en.wikipedia.org/w). Ich korekta nie trwała dłużej i kosztowała więcej niż tradycyjne tłumaczenie tych samych dokumentów. Stwierdzono wtedy, iż bardzo mało prawdopodobne jest uzyskanie szybkiego tłumaczenia maszynowego [[27]](#_[Online]_https://www.masswerk.at/el). Dany raport doprowadził do odcięcia funduszy od rządu USA i wywarło to bardzo negatywny wpływ na badania związane z NLP do tłumaczenia języka maszynowego. Jednakże pomimo problemów z dotacjami, badania nad NLP nie zostały całkowicie porzucone [[4]](#_Y._Bar-Hillel,_"Appendix).

Kilka prototypów systemów dotyczących przetwarzania języka naturalnego zostało rozwiniętych w następnym dziesięcioleciu. W 1966 roku niemiecki amerykański naukowiec i pracownik MIT Joseph Weizenbaum stworzył program symulujący psychoanalitykę o nazwie ELIZA. Dla zadanych słów kluczowych w zdaniach wejściowych, zadawał danemu pacjentowi pytania na ich temat i zachęcił go do dalszego wyjaśnienia tematu. Mimo prostoty pomysłu wystarczyło zachować u danego pacjenta poczucie, iż mówca jest słyszalny i rozumiany. Twórca ELIZY uważał także program za metodę ukazania powierzchowności komunikacji pomiędzy człowiekiem a maszyną. Wielu naukowców było przeświadczonych, iż program będzie w stanie wpłynąć pozytywnie na życie i relacje wielu ludzi, którzy cierpią ze względu na problemy psychologiczne, jak i przysłuży się lekarzom pracującym nad leczeniem takich pacjentów. ELIZA miała zaimplementowany w sobie skrypt DOKTOR, który miał stanowić, „odpowiedzi bezkierunkowego psychoterapeuty na wstępny wywiad psychiatryczny”, co oznacza, iż miała być to próba komunikacji między człowiekiem a pacjentem. Wzorce mowy miała wynikać z dostarczonego skryptu, a sam tekst był badany pod kątem występowania słów kluczowych, ustawiania ich w kolejności oraz organizacji reguł transformacji dla wyjścia końcowego. Algorytmy DOCTORa pozwoliły na zwodniczą inteligentną reakcję programu, która była w stanie oszukać wiele osób o swojej „inteligentnej” reakcji przy pierwszym użyciu programu [[33]](#_[Online]_https://en.wikipedia.org/w_6).



Rys. 3.1 Joseph Weizenbaum przy programie ELIZA. (https://www.masswerk.at/elizabot/)

W 1970 roku twórca programu SHRDLU Terry Winograd rozwinął swój system na potrzeby pracy doktorskiej. Był w stanie przetwarzać polecenia użytkownika w języku naturalnym do manipulacji wirtualnymi blokami i odpowiedzi na ich pytania. W latach 80. NLP zaczęło wychodzić z 14-letniego „bezruchu”. Wzrost mocy obliczeniowej komputerów, który wystąpił w danym czasie pozwalał naukowcom spróbować nowych środków, takich jak uczenie maszynowe. Wczesne próby wykorzystania algorytmów z tej dziedziny w postaci drzew decyzyjnych generowały złożone reguły, które znacznie różniły się od dotychczasowych. Zmieniło się to, gdy podejście językowe zaczęło tracić na znaczeniu na rzecz statystyk. Były to obiecujące wyniki, które dawały zastosowanie w modelach statystycznych podejmowania probabilistycznych decyzji na podstawie przypisywanych wag dla danych wejściowych tekstu. Takie modele uzyskały zastosowanie w dziedzinie algorytmów uczenia maszynowego na dużych tekstach z adnotacjami zwanymi też „korpusami”. Modele mogły specjalizować się, aby lepiej pasować do jednej kategorii. Ta metoda była korzystna, dostarczała mniejszej liczby szerszych zasad zamiast wielu szczegółów i była lepsza w obsłudze nieznanych lub błędnych danych wejściowych [[6]](#_[Online]_https://pl.wikipedia.org/w).

Popularność statystycznego NLP rosła przez lata 90. równolegle z rosnącą liczbą zasobów tekstowych, które zaczęły być publikowane w Internecie. Dostarczało to cennych danych do modelu statystycznego jako zbioru treningowego. W ciągu całej dekady zaczęły pojawiać się chatboty. W roku 1997 projekt Jabberwacky został ukończony i finalnie uruchomiony w Internecie. Jego wynik był pewnego rodzaju chatbotem, który miał na celu symulowanie języka naturalnego w humorystyczny dla użytkownik sposób. Potrafił nauczyć się nowych słów, żartów lub innych cech językowych, na podstawie wcześniej prowadzonych rozmów, które były przechowywane przez program [[8]](#_[Online]_https://en.wikipedia.org/w_2).

W XXI wieku NLP rozwija się dalej dzięki dostępności tanich i wydajnych komputerów, dużego zapotrzebowania oraz zastosowaniu modelu big data. Modele rekurencyjnych sieci neuronowych wprowadzone zostały w latach 90. Przez kolejne lata okazały się bardzo przydane w przetwarzaniu głosu i symulowaniu jego. W 2011 roku firma Apple przedstawiła pierwsze urządzenie oparte na sztucznej inteligencji Siri jako asystent głosowy udostępnione klientom kupującym urządzenia tej marki [[9]](#_[Online]_https://pl.wikipedia.org/w_1).

Dziś NLP jest częścią życia każdego człowieka. Tłumaczenie maszynowe jest wbudowane często jako funkcja wyszukiwarek internetowych lub mediów społecznościowych. Strony internetowe posiadają też modele wirtualnych asystentów w postaci chatbotów, które często udzielają odpowiedzi na problemy lub często zadawane pytania.

## Podstawowe operacje

Pierwszym krokiem w kierunku przetwarzania tekstu do języka naturalnego jest segmentacja. Dany tekst jest podzielony na pojedyncze, możliwe do zidentyfikowania jednostki językowe, które stanowią część danych. Taka operacja nazywana jest tokenizacją. Zdecydowanie większość tokenów jest podzielona na wyrazy, uzyskane poprzez podzielenie tekstu spacjami lub odrzucenie danych znaków interpunkcyjnych. Istnieją jednak ważne reguły, które należy przestrzegać. W języku angielskim nieformalne negacje są pisane wspólnie z pomocniczymi. Niektóre metodologie tokenizacji mogą traktować dane słowo jako oddzielne dwa tokeny, które zatwierdzają różne funkcjonalności. Przykładowo zwrot „nie musiał” zostanie podzielone na „musiał” jako słowo pomocnicze, wskazujący czas oraz negację „nie” jako przysłówek wskazujący na zaprzeczenie. Punkt podziału wtedy nie jest ani interpunkcją, ani białą spacją. Istnieją różne wyjątki dotyczące identyfikatorów i nazw własnych. Zadanie tokenizacji jest bardzo zróżnicowane w zależności od stosowanego języka i analizowanego tekstu. W języku włoskim dany przedimek określony może być powiązane z apostrofem ze słowem, które poprzedza w danej chwili. Język niemiecki charakteryzuje się długimi słowami, które należy oddzielić w osobnej analizie. W języku japońskim bądź chińskim nie używa się spacji pomiędzy słowami i należy wtedy zastosować inne zasady segmentacji specyficzne dla danego języka [[13]](#_[Online]_https://tokeneo.com/pl/czy).

Drugim zadaniem jakie musi spełniać segmentacja jest podzielenie tekstu na zdania. Znalezienie ich utrudniają problemy podobne w przypadku tokenizacji. W zależności od podejścia, cytaty w zdaniach i po obu stronach można traktować jako oddzielne zdania [[10]](#_[Online]_https://en.wikipedia.org/w_1).

Kolejnym krokiem jest normalizacja. Podczas przetwarzania danych tokenów, powinny być one traktowane w ten sam sposób pod względem semantycznym, mimo iż reprezentujące je znaki są różne. Jednym z takich przykładów są wielkie litery na początku zdań. Do podstawowych czynności jakie w tym celu wykonać należy składanie i usuwanie znaków interpunkcyjnych. Analizowane słowa mogą mieć różne zakończenia mimo tych samych znaczeń. Występuje to wielokrotnie przez zastosowanie liczby mnogiej rzeczowników bądź różnych odmian czasowników. Występuje tutaj operacja nazywana procesem heurystycznym. Nie ma on uniwersalnej definicji, ale bazuje na procesie usuwania afiksów i tworzenia bazy danej ze słowa zwanego rdzeniem. Stemming określa użycie różnych końcówek dla danego słowa. Najlepszym algorytmem rdzeniowym był algorytm Portera, zastosowany dla tekstów w języku angielskim. Został on opracowany przez Martina Portera w 1979 roku. Dzięki prostej implementacji opiera się on na stopniowym usuwaniu i zastępowaniu niektórych liter na końcu słowa. W pięciu krokach ostatnie litery są usuwane i zastępowane w celi pozbycia się zakłóceń i uzyskaniu odpowiedniej formy bazowej. W zależności od ułożenia samogłosek lub spółgłosek oraz długości słowa, dane kroki są wykonywane lub pomijane. Niestety niektóre zmiany wprowadzone przez algorytm mogą spowodować zapadnięcie się słowa w wspólną formę bazową, ale o odmiennym znaczeniu [[12]](#_[Online]_https://pl.wikipedia.org/w_2). Czasami lepiej zachować tylko podstawową formę odmiennego słowa i ograniczyć redukcję, aby zachować daną część mowy. Innym razem lepiej pobierać informacje obojętne na daną część wypowiedzi i jednakowo traktować całość.

Bardziej złożony i drobiazgowy proces, którego cel jest taki sam jak steemingu to lematyzacja. W odróżnieniu do wprowadzania słowa, otrzymany wynik jest zawsze prawdziwy. Posługuje się ona głębokim słownictwem i morfologiczną analizą, zwykle z wykorzystaniem danej bazy leksykalnej. Aby uzyskać lepsza wydajność, stosuje się dodatkowe informacje zawarte w kontekście wypowiedzi takie jak jej część albo analizowane słowo. Zasadniczą wadą lematyzatorów jest wolniejsze działanie i nieobsługiwanie nowych oraz niestandardowych i wymyślonych słów, które nie występują w bazie danych używanych przez algorytm [[11]](#_[Online]_https://en.wikipedia.org/w_3).

Następnym krokiem jest znakowanie części mowy, z użyciem tzw. znaczników POS. Jest to proces przypisywania do każdego tokena etykiety, która odpowiada danej części mowy. Liczba przypisywanych znaczników może być też większa niż liczba części mowy występująca w analizowanym tekście i różnić się od znaczników POS. W zależności od szczegółowości podkategorii części mowy są przypisywane przez system. Znaczniki mogą się różnić podstawową i przeszłą formą czasowników lub nawet używać odpowiedniego znacznika dla bezokoliczników. Pomimo braku uniwersalnego zestawu znaczników istnieje wiele bardzo popularnych, które są powszechnie używane. Jeśli chodzi o użyty w aplikacji język angielski jest on zdefiniowany jako zestaw znaczników Brown opracowany w 1979 roku na Uniwersytecie Brown zawierający 87 znaczników. Najczęściej używany jest zestaw Penn Treebank, który został uproszczony do 48 znaczników i opracowany na Uniwersytecie Pensylwanii w latach 90. Oba zestawy były częścią projektów opisywania dużej ilości kolekcji „korpusów”. Najważniejszą częścią oznakowania opisanego na początku rozdziału jest ujednoznacznienie w przypadku słów posiadających wiele znaczeń. Można wyróżnić dwa podejścia oparte na regułach oraz znakowanie POS. Były wykorzystane predefiniowane reguły do analizy morfologii analizowanego słowa oraz słów sąsiednich, aby określić jego przynależność do danej kategorii. Jeden z najstarszych oraz bardzo popularnych sposobów znakowania POS i opartym na jego regułach jest tagger firmy Brill. Został przeszkolony na „korpusach”, których tokeny zostały już ręcznie oznaczone i określają zasady przypisywania do danej części mowy. Podejście stochastyczne określa znacznik na podstawie jego częstotliwości oraz prawdopodobieństwa. Najbardziej podstawową metodą jest przypisywanie słowom części mowy, która jest niejednoznaczna. Inna metoda jest oparta na n-gramowym podejściu. Kiedy pojawia się słowo, które może być ujednoznacznienie, określany jest przypisany do niego znacznik, wybierając najbardziej prawdopodobny po wprowadzeniu n poprzedzających znaczników na podstawie sekwencji n kolejnych w danych uczących. Najbardziej dokładny wynik uzyskuje się stosując model Markowa, który uwzględnia zarówno prawdopodobieństwo kolejności znaczników oraz prawdopodobieństwo przypisania jednego z możliwych znaczników do słowa [[14]](#_M._Marcus,_S.).

Następnym krokiem NLP jest określenie struktury zdań i zależności między słowami. Proces wprowadzenia danej struktury składniowej dla sekwencji słów nazywana jest analizowaniem (ang. parsing). Istnieją trzy różne typy analizowania: Shallow parsing, analiza okręgów wyborczych oraz analiza zależności. Pierwszy typ to technika grupowania tokenów w frazy, które mają tą samą funkcję co część mowy ich słowa głównego. Każda fraza składa się z głównego słowa i innych słów, które go opisują [[15]](#_[Online]_https://en.wikipedia.org/w_4). Następnym typem, czyli analizą okręgów jest podejście modelowania hierarchicznej struktury zdania. Parser przetwarza zdanie za pomocą gramatyki formalnej, czyli zbioru reguł i definiuje możliwe konstrukcje w języku. Przedstawia wyniki w postaci struktury drzewa zwanego parse tree. Ostatnim typem, czyli analizą zależności jest identyfikowanie i nazywanie relacji w zdaniu. Każde słowo ma przypisywane inne słowo, z zdania które modyfikuje, zwanego też jego głową. Parser dostaje adnotacje do każdej takiej relacji ze znacznikiem zależności, który opisuje jej charakter [[39]](#_[Online]_https://towardsdatascience).

Ostatnim krokiem jest analiza nastrojów (ang. sentiment analysis). Jest to proces określenia opinii na dany temat w tekście. Identyfikuje wartości sentymentu, które można przeprowadzić na różnych poziomach oceniając uczucia wyrażane przez poszczególne słowa, zdania, fragmenty lub cały dokument. Wyróżniane są dwa rodzaje systemów analizy nastrojów: oparty na regułach i automatyczny [[16]](#_[Online]_https://ichi.pro/pl/analiz).

Pierwszy opiera się w dużej mierze na leksykonach, czyli bibliotekach słów z ręcznie przypisaną wartością sentymentu. Oznaczone są punktacją za każdą biegunowość, czyli określony atrybut, czy dan termin ma opinię pozytywną czy negatywną. Drugie systemy wykorzystują podejście języka maszynowego które wymaga przeszkolenia, danego modelu klasyfikatora na dużej liczbie wystąpień oznakowanego dokumentu. Jego zaletą jest przewidywanie z większym prawdopodobieństwem uczuć związanych z nowymi wyrażeniami bądź slangiem [[16]](#_[Online]_https://ichi.pro/pl/analiz).

## Podobne rozwiązania

Stale pojawiają się nowe rozwiązania problemu wyciągania przydatnych informacji z dużej ilości tekstu. Jednym z przykładów może być Amazon Textract, która jest usługą uczenia maszynowego, wyodrębniająca tekst, pismo odręczne i dane skanowanych dokumentów. Obecnie wiele firm ręcznie wyodrębnia dane z formularzy i tabel. Omawiane rozwiązanie kładzie nacisk na zeskanowane dokumenty [[17]](#_[Online]_https://aws.amazon.com/tex). Może to służyć szybkiemu zautomatyzowaniu przetwarzania dokumentów i podejmować działanie na uzyskanych informacjach. Dane są bardzo szybko wyodrębniane i przetwarzane.

Użycie NLP jest wykorzystane w tłumaczeniu dużych tekstów w Internecie. Bardzo szybka analiza znaczeniowa jest też ważna w przypadku właściwego przetłumaczeniu teksu na danym portalu internetowym, aby nie wprowadzić użytkownika w błąd. Jest wykorzystane także w celu asystentów cyfrowych dedykowanych dla potencjalnych marek. Doradzają oni i odpowiadają na pytania danego użytkownika w danej dziedzinie. Przykładem jest firma Apple, która stworzyła Siri jako asystenta dla każdego konsumenta z systemem dostarczonym przez tą firmę [[9]](#_[Online]_https://pl.wikipedia.org/w_1).

Twiggle wdraża jedne z najbardziej zaawansowanych rozwiązań w dziedzinie nauczania maszynowego, sztucznej inteligencji oraz Natural Language Processing. Stworzona jest, aby wspierać eCommerce, które jest coraz bardziej odpowiednie oraz angażujące dla klientów. Firma skupia się, aby dotrzeć do nich i do sedna interakcji z nimi. Oferuje rozwiązania dla technologii wyszukiwania opartych na interakcjach między ludzkich oraz dokładnego zrozumienia struktury językowej. Boty mogą naśladować zachowanie prawdziwych sprzedawców [[18]](#_[Online]_https://hbsp.harvard.edu/p). Zrozumieć czego potrzebują klienci bez względu na to o co pytają.

Firma TaskUs dostarcza rozwiązania związane z obsługą klienta nowej generacji. Opiera się na zrozumieniu języka naturalnego i dostarcza narzędzie do jego przetwarzania. Oferta firmy wykorzystuje zautomatyzowane procesy, aby dostarczyć rozwiązania klientom szybciej niż kiedykolwiek wcześniej. TaskUs wyróżnia się także możliwością szybkiego skalowania [[19]](#_[Online]_https://ir.taskus.com/).

Wykorzystanie NLP stosuje się także w dziale zarządzania zasobami ludzkimi (HR), który jest niekiedy żmudnym, powolnym i często niedokładnym zadaniem. Oferuje ono zestaw narzędzi do analizy nastrojów ludzkich, identyfikacji potencjału i improwizacji kompetencji. Może być wykorzystane w przetwarzaniu listów motywacyjnych, danych osobowych oraz CV. Minimalizuje też ludzkie uprzedzenia w podejmowaniu decyzji [[20]](#_[Online]_https://em360tech.com/top-).

Kolejną dziedziną wykorzystania przetwarzania języka naturalnego jest system opieki zdrowotnej. Aplikacja MedInReal redukuje trudności związane z poświęceniem odpowiedniej ilości czasu danemu pacjentowi. Asystenci cyfrowi przejmują papierkowe obowiązki, z którymi do czynienia mają przeważnie lekarze [[21]](#_[Online]_https://medinreal.com/). Ułatwiają też komunikację na temat ich leczenia i starają się zrozumieć ich potrzeby.

Analiza dużych zbiorów danych (Big Data) opiera się na wykorzystaniu NLP poprzez wyszukiwanie danych fraz i informacji w dużej kolekcji zbiorów tekstowych. Szybszy i dokładniejszy dostęp do informacji przyśpiesza wszystkie inne procesy za pomocą wyszukiwania, używając zapytań własnymi słowami, dzięki temu użytkownicy nie są już ograniczeni wyborem lub znajomością „właściwych” słów kluczowych [[20]](#_[Online]_https://em360tech.com/top-). Analiza tekstów jest bardziej skoncentrowana na ich przeszukiwaniu.

Generative Pre-trained Transformer 3 (GPT-3) definiuje autoregresyjny model języka, który poprzez zastosowanie głębokiego uczenia, tworzy tekst możliwy do zrozumienia przez człowieka. Został stworzony przez OpenAI w laboratorium sztucznej inteligencji w San Franscisco w 2020 roku. Stanowi część trendu występującego w systemach przetwarzania języka naturalnego NLP. Jakość tekstu generowana przez dany model jest tak wysoka, że wielokrotnie jest go ciężko odróżnić od tekstu napisanego przez człowieka, co niesie za sobą korzyści jak i tworzy potencjalne ryzyko. W artykule twórcy języka ostrzegli przez skutkami ubocznymi stosowania tego modelu. Firma Mircosoft wraz z 22 września 2020 roku ogłosiła, że ma dostęp do podstawowego kodu GPT-3, natomiast nadal udziela publicznych licencji do stosowania go [[34]](#_https://en.wikipedia.org/wiki/GPT-3).

Jednym z rozwiązań bazującym na inteligentnym przetwarzaniu tekstu jest GitHub Copilot, zasilany przez Codex, stworzony jako asystent kodowania w środowiskach programistycznych. Syntetyzuje on kod oraz używa podobnego kontekstu, aby dopasować go do reszty dokumentu. Jest w stanie przetworzyć komentarze w kod według opisanej w nich logiki. Może powielać schematy kodu zawarte według podanego wzorca. Jest w stanie zasugerować testy dla danego systemu, a przede wszystkim pokazuje listę potencjalnych rozwiązań dla danych linijek kodu [[35]](#_[Online]_https://copilot.github.com).

# Opis środowiska i aplikacji

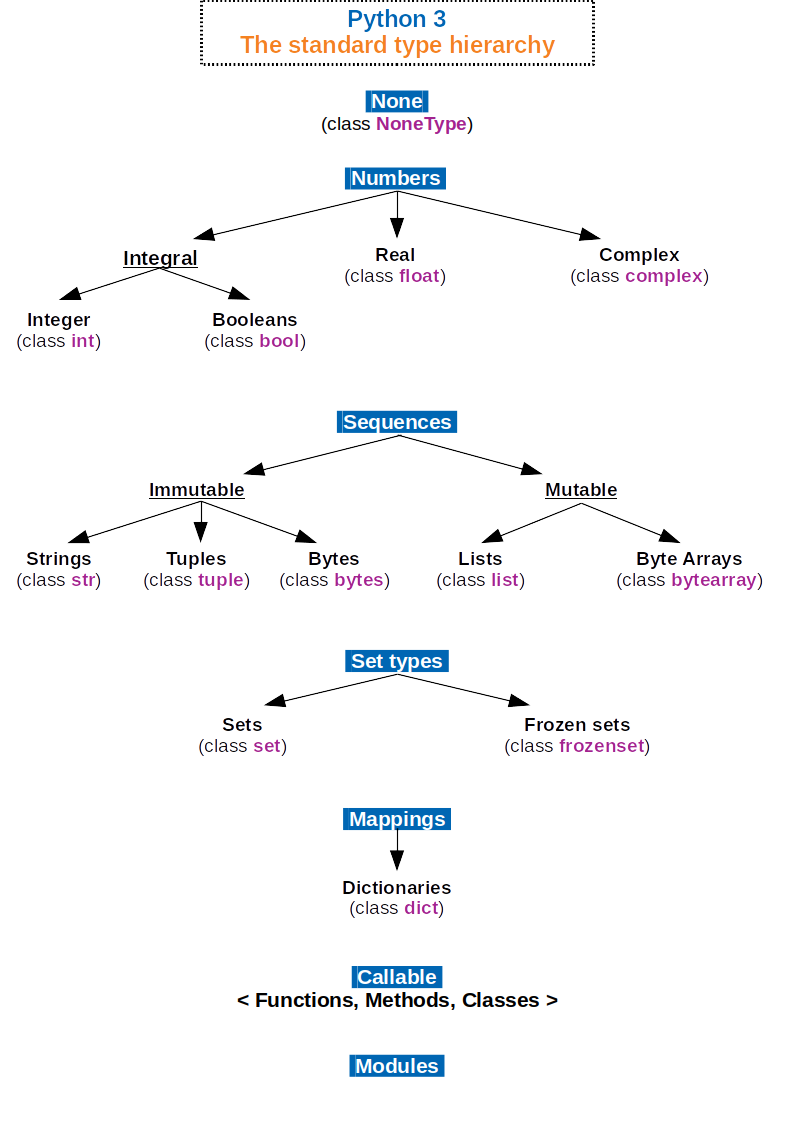
Do stworzenia aplikacji Machine Book Extract, opisanej w dalszej części pracy, został użyty język Python w wersji 3.7.9 oraz trzy biblioteki: spaCy 3.0.5, pyqt w wersji 5.0 i pandas 1.3.0. Całość kodu była przetrzymywana na zdalnym, prywatnym repozytorium. Dany język oraz biblioteki zostały wykorzystane z powodu swojego przeznaczenia związanego z przetwarzaniem języka naturalnego, interfejsem użytkownika i łatwym zarządzaniem dużą ilością danych. Poniższa aplikacja jest otwarta na rozszerzenia oraz nowe funkcjonalności. Jest natomiast zamknięta na modyfikacje. W celu ergonomii ekranu danego użytkownika użyto środowiska aplikacji okienkowych. Program został pokryty testami oraz wykorzystany do wstępnej analizy danych.

## Język Python

Python to język wysokiego poziomu o wysoce rozbudowanym pakiecie bibliotek standardowych. Jego przewodnią ideą jest klarowność oraz czytelność kodu. Cechuje się zwięzłością i przejrzystością. Pyton wspiera także różne paradygmaty programowania. Między innymi imperatywny i obiektowy. W mniejszym stopniu funkcyjny. Posiada pełen system typowania, który jest także dynamiczny oraz potrafi automatycznie zarządzać pamięcią. Jest podobny do Perl i Ruby pod tym względem. Jest bardzo często używany jako język skryptowy i jest dostępny na wielu systemach operacyjnych.

Język Python powstał w latach 90. jako następca ABC – języka stworzonego w Centrum Matematyki i Informatyki w Amsterdamie. Jego twórcą jest Guido van Rossum, a nazwa języka, mimo iż często mylona nie pochodzi od nazwy zwierzęcia, ale od serialu komediowego, który był emitowany w latach 70. na stacji BBC o nazwie „Latający Cyrk Monthy Pythona”. Projektant bowiem był wielkim fanem tego serialu.

Filozofia Pythona opiera się na kilku paradygmatach. Podobnie do języka C++ nie wymusza jednego stylu w jakim dany użytkownik może programować. Możliwe w tym języku jest także programowanie obiektowe, strukturalne i funkcyjne. Typy są sprawdzane dynamicznie, a za zarządzanie pamięcią odpowiada odśmiecacz pamięci (ang. garbage collection). Hierarchia typów w języku Python została przedstawiona na Rys. 4.1.



Rys. 4.1 Hierarchia typów obecna w języku Python. (https://commons.wikimedia.org/wiki/File:Python\_3.\_The\_standard\_type\_hierarchy.png)

Python ponadto operuje na systemie typów, w których wszystko jest obiektem. Dana klasa może dziedziczyć z dowolnego typu. Można zatem dziedziczyć z napisów czy słowników. Ponadto obecne jest też dziedziczenie wielokrotne, podobnie jak w języku C++. Typy można odczytywać i porównywać, a atrybuty obiektu mogą być też pobierane przez użytkownika. Każda klasa może być opatrzona dynamiczną dokumentacją w kodzie źródłowym, co znacznie poprawia jakość zrozumienia kodu. Język Python posiada także bardzo dobrze rozwiniętą bibliotekę standardową przeznaczoną do wykonywania wielu zadań. Metody danej biblioteki można uzupełnić modułami napisanymi w C lub Pythonie. Jest on też szczególnie dostosowany do tworzenia systemów sieciowych, jako iż obsługuje dużą ilość protokołów. Dostarczone moduły są w stanie tworzyć systemy obsługujące interfejsy użytkownika, obróbki wyrażeń regularnych oraz są skierowane na serwery WWW. Biblioteki mogą być też stosowane na wielu platformach, dzięki czemu większe aplikacje są uruchamiane na dużej ilości dedykowanych systemów operacyjnych [[22]](#_[Online]_https://pl.wikipedia.org/w_3).

Interpreter Pythona posiada też tryb interaktywny. Mona tam wprowadzać wyrażenia z terminala i otrzymywać wyniki. Może to ułatwić naukę programowania we wczesnym jej etapie. Standardowy moduł nie jest jednak zbyt wygodny i braku wielu w nim funkcji. Przeznaczoną biblioteką do testów w języku Python jest unittest, który zostanie szerzej opisany w dalszych rozdziałach [[23]](#_[Online]_https://docs.python.org/3/).

## Biblioteka spaCy

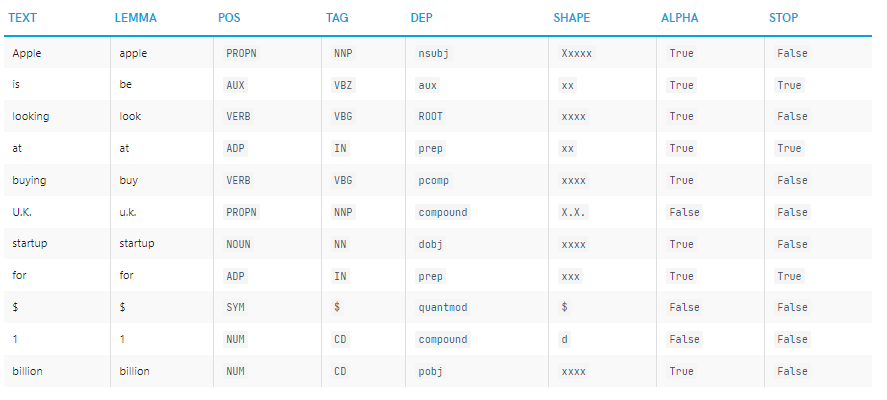
Biblioteka spaCy jest używany przede wszystkim w wykorzystaniu NLP do celów programistycznych. Wspierany jest przez ponad 64 języki. Wykorzystuje wielozadaniowe uczenie się z wstępnie wytrenowanymi transformacjami oraz asynchroniczność w przetwarzaniu słów i zdań. Tokenizacja w nim jest mocno motywowana językowo. Stworzone zostały specjalne komponenty do rozpoznawania jednostek słów, zależności, oznaczania części mowy, analizy morfologicznej, łączenia jednostek i lematyzacji. Jest łatwo rozszerzalny dzięki atrybutom i niestandardowym komponentom. Ponadto biblioteka wprowadza kompleksowy oraz rozszerzalny system konfigurujący tzw. „training runs”. Plik konfiguracyjny będzie opisywał każdy szczegół takiego przebiegu treningowego, bez ukrywania wartości, które są domyślne oraz ułatwi opanowanie przeprowadzania doświadczeń i eksperymentów, śledząc przy tym każde zmiany.

Płynna ścieżka od prototypu do produkcji pozwala śledzić każdy etap transformacji, wewnętrznego przetwarzania oraz szkolenia danych dając pewność, iż projekt zawsze będzie w stanie być przekazany do automatyzacji. Oferuje także pobieranie zasobów źródłowych, weryfikację sum kontrolnych, wykonywanie poleceń, ale także możliwość buforowania z różnymi integracjami i systemami backendowymi.

SpaCy wprowadza także ścieżki przepływu oparte na transformacjach, które zapewniają także, iż dokładność operacji jest zależna od obecnego stanu algorytmów NLP. Można użyć go pod kątem optymalizacji procesora, w różnego rodzaju algorytmach. Sama biblioteka ma wiele preferowanych rozszerzeń dla różnych języków, od angielskiego -najpopularniejszego, po różne inne języki światowe (np. niemiecki, polski, słowacki). Dodatek en\_core\_web\_sm zapewnia integrację z biblioteką spaCy oraz zapewnia dodatkowe funkcjonalności takie jak:

* Obsługiwany typ języka
* Typ: słownictwo, składnia, encje i wektory
* Gatunek: tekst pisany (blogi, newsy, komentarze)
* Wielkość rozszerzenia
* Składniki: token2vec, parser, tagger, nadawca, lematyzator, ner, atrybut
* Ścieżkę przepływu: tok2vec, tagger, parser, ner, lematyzator, atrybut
* Wektory
* Źródła

Ponadto spaCy zawiera tagger, umożliwiający nadanie każdemu ze znalezionych słów jego unikalnego znacznika, umożliwiającego zdefiniowanie słowa jako części gramatycznej w zdaniu. W zaawansowanych przypadkach daje możliwość zdefiniowania, czy dane słowo jest nazwą osoby, miejsca, numeru lub związane z czasem. Przykładowy rozkład zdania „Apple is looking at buying U.K. startup for $1 billion” w spaCy został przedstawiony na rysunku 4.2.

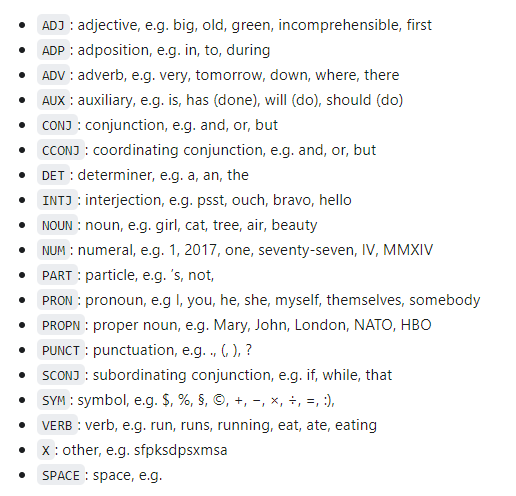


Rys. 4.2 Przykład rozbioru zdania „Apple is looking at buying U.K. startup for $1 billion” przez bibliotekę spaCy (https://spacy.io/usage/linguistic-features)

Zdanie zostało rozbite na wiele atrybutów oznaczające:

* TEXT – słowo w zdaniu
* LEMMA – jako podstawowa forma słowa, dla czasowników bezokolicznik
* POS – część mowy danego słowa
* TAG – jako znacznik części mowy
* DEP – zależność syntaktyczna (czyli relacja między tokenami)
* SHAPE – kształt słowa (np. wielkość liter i interpunkcja, format cyfr)
* ALPHA – czy token jest znakiem alfanumerycznym
* STOP – czy słowo jest słowem zatrzymania, czyli słowem o małym znaczeniu (spójnikiem) lub słowem popularnym, nie wpływającym na identyfikację dokumentu, czy częścią listy zatrzymania, czyli listą takich słów

SpaCy zapewnia też części mowy widoczne na Rys. 4.3. Każda z części mowy może mieć końcówki, definiującą znacznik z innym rodzajem końcówki. Dlatego przy zbiorczej analizie części mowy dla danego tekstu, taki przypadek musi zostać uwzględniony w aplikacjach i algorytmach.



Rys. 4.3 Przykładowe znaczniki części mowy używane przez bibliotekę spaCy (https://newbedev.com/what-do-spacy-s-part-of-speech-and-dependency-tags-mean)

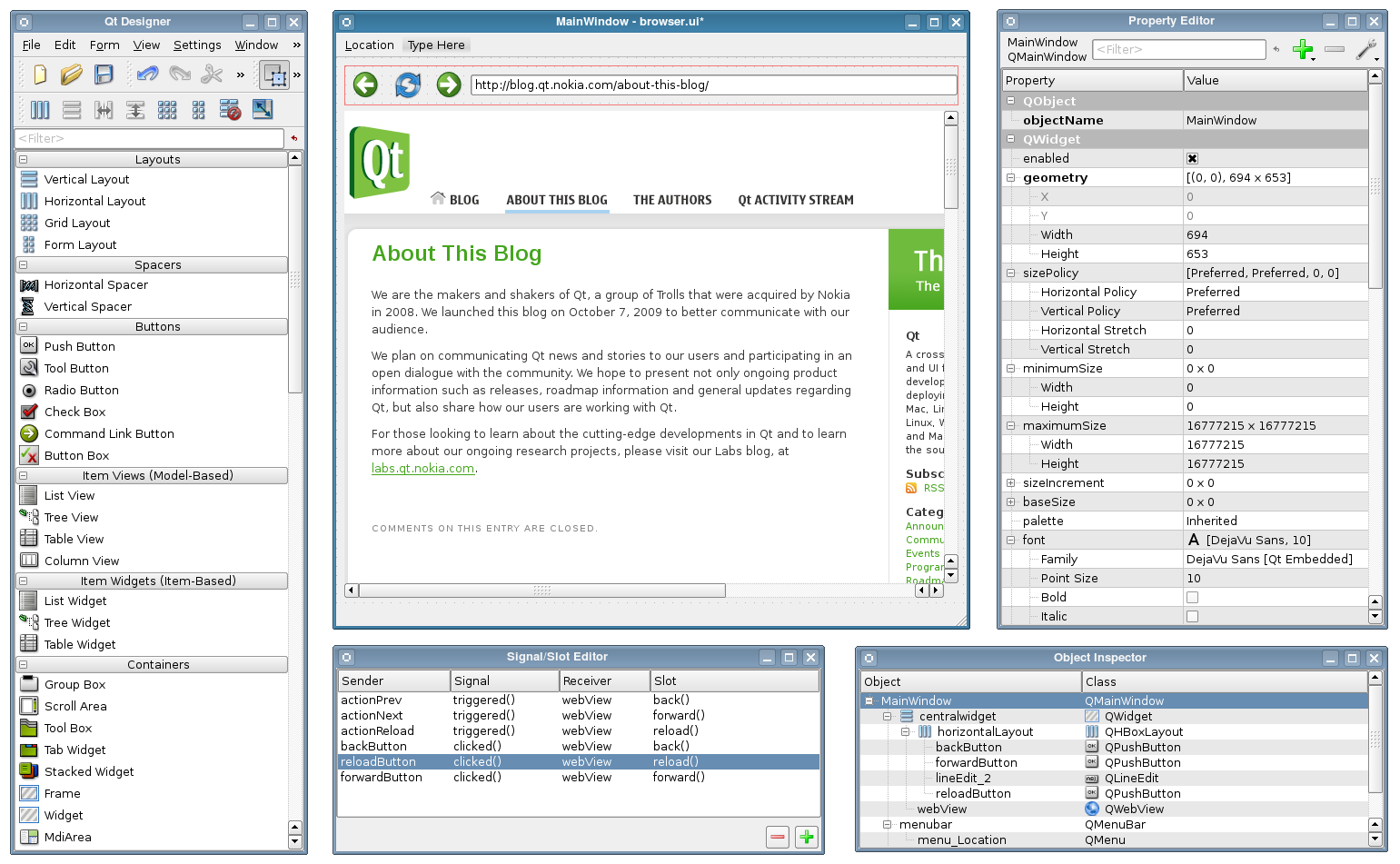
SpaCy posiada także składnik potoku, umożliwiający wielojęzyczne adnotacje sylab dla danych tokenów. Używa do tego modułu Pyphen i obsługuje daną listę języków. Jest to niezwykle przydatne przy mechanizmach mierzenia czytelności danego tekstu opisanych w dalszych rozdziałach [[24]](#_[Online]_https://spacy.io/usage/spa).

## Biblioteka pyqt

Biblioteka PyQt jest jednym z najbardziej popularnych narzędzi języka Python. Został opracowany przez Riverbank Computing Limited, a sam jest rozwijany w ramach projektu Qt. Zapewnia powiązania dla różnych wersji, darmowych jak i komercyjnych. Sama biblioteka umożliwia tworzenie interfejsów graficznych PyQt, ale również obsługuje powiązania z różnymi bazami danych, zarządzanie nimi i tworzeniem prostej grafiki. Nakładka jest dostępna dla systemów operacyjnych MS Windows, Linux, Unix jak i macOS X. Wiele systemów wbudowanych także wykorzystuje daną nakładkę, gdzie jej najnowsza wersja obsługuje wersję PqQt5 [[25]](#_[Online]_https://pypi.org/project/P).

Proste tworzenie GUI zapewnia powiązany z frameworkiem PyQt – Qt Creator. Jest to wieloplatformowe środowisko dla programistów. Zawiera graficzny interfejs dla debuggera oraz wiele narzędzi do projektowania różnych interfejsów graficznych, wedle upodobań użytkowników. Dzięki niemu można w łatwy sposób zmienić wygląd aplikacji, bez zmieniania logiki wewnętrznej. Dany kreator został użyty podczas pisania aplikacji Machine Extract Book, w celu zapewnienia przejrzystego interfejsu dla użytkownika.

Qt Creator zawiera w sobie Qt Designer, który jest narzędziem do projektowania i budowy interfejsu programów za pomocą widżetów. Jest zintegrowany z biblioteką Qt i udostępnia możliwość testowania zachowania interfejsu zawierającego różne style i rozdzielczości. Elementy są automatycznie tworzone i integrowane z kodem źródłowym poprzez mechanizmy sygnałów i slotów [[26]](#_[Online]_https://www.qt.io/product). Przykładowy wygląd Qt Designer został przedstawiony na Rys. 4.4.



Rys. 4.4 Wygląd aplikacji Qt Designer dla tworzenia aplikacji okienkowej (https://pl.wikipedia.org/wiki/PyQt#/media/Plik:Screenshot-qt-designer-qwebview.png)

# Opis metryk

Utwór literacki jest tekstem słownym, pisanym, ale też i ustnym, ukształtowanym z środków jakimi dysponuje język, na sposób artystyczny. Jest często odmienny od dzieła naukowego i użytkowego. Teksty spoza literatury pięknej, spełniające niekiedy cele użytkowe składają się na literaturę stosowaną. Takie utwory posiadają wiele atrybutów pomocnych przy zdefiniowaniu celu napisania wraz z niosącym przez daną książkę przesłaniem lub morałem. Czytelność danego tekstu pozwala na zdefiniowanie odpowiedniej grupy jego odbiorców. W tym rozdziale prócz opisu atrybutów tekstu i stylów zostało przedstawione porównanie otrzymanych wyników na przykładzie dwóch gatunków literackich.

## Atrybuty długich teksów pisanych

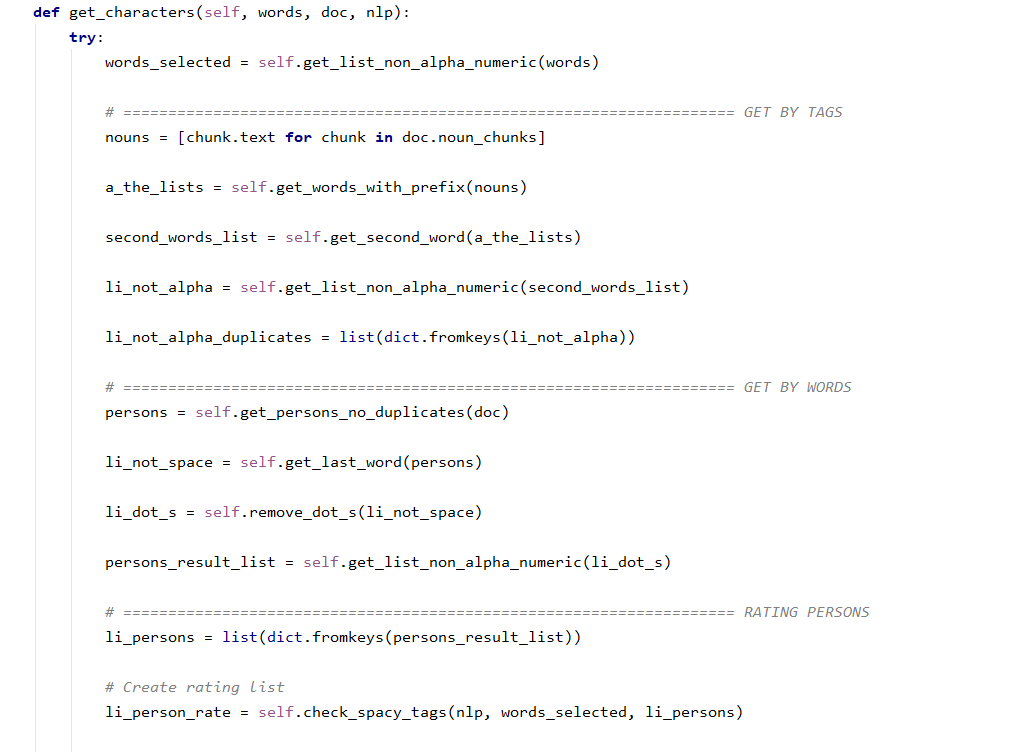
Wiele lektur szkolnych posiada opracowania, które ułatwiają analizę tekstu oraz pisanie rozprawek na jego podstawie, bądź poruszanie problematyki w nim zawartej. Wyciągnięcie najważniejszych informacji zawartych w danych tekstach jest niezwykle trudne, nawet przy dobrej znajomości całego tekstu. Wyciągnięcie przykładowych metryk może być pomocne przy dalszej analizie i znacząco skrócić jej czas. Każda książka posiada wiele atrybutów, które definiują czytany tekst.

Geneza powstania utworu określa rok, miejsce i autora danego dzieła. Wyciągnięcie takich informacji z tekstu jest możliwe, aczkolwiek wiąże się z biografią autora, która, jeśli nie jest uwzględniona w tekście nie daje szansy zdefiniowania genezy powstania utworu [[28]](#_[Online]_https://klp.pl/syzyfowe-pr).

Liczba słów, znaków, zdań a także średnia liczba słów w zdaniu i znaków w danym tekście często definiuje szybkość czytania danego tekstu, co pozwala na oszacowanie jak długo można czytać dany utwór, czy też ile lektur można przeczytać w danym czasie. Przy trudniejszych książkach o bardziej zaawansowanym słownictwie takie wartości mogą być kluczowe, aby poświęcić więcej czasu ku dokładniejszej analizie danego dzieła literackiego. Takie wartości mogą być też ważne dla czytelności danego tekstu oraz skierowaniu go do odpowiedniej grupy wiekowej czy też grupy uczącej się danego języka na odpowiednim stopniu zaawansowania. Miara czytelności tekstu może być przydatna przy klasyfikacji tekstów w języku obcym (dla danego czytelnika), tj. może pomóc w podziale tekstów na: dla początkujących, średnio zaawansowanych itd.

Każda książka zawiera czas i miejsce akcji utworu. Sytuacja jest tu podobna, ale wyciągnięcie informacji jest już zależne od danego tekstu. Niekiedy nie jest bezpośrednio podany czas akcji danego utworu, jedynie można wywnioskować go poprzez opisy natury, ale także i genezę danego dzieła. Są to wartości powiązane ze sobą. Miejsce akcji też może być zdefiniowane poprzez opisy i dodatkowe informacje zawarte w tekście. Trzeba jednak zauważyć, że próba wyciągnięcia takich informacji została uwzględniona w opisywanej pracy oraz niosące z nią konsekwencje. Liczba czasowników, w czasie teraźniejszym i przeszłym jest jednym ze wskaźników, które mogą pomóc przy zdefiniowaniu czasu akcji, czy dane wydarzenia toczą się w teraźniejszości czy przeszłości. W celu genezy danego dzieła można zobaczyć, czy książka ma charakter retrospekcji czy dzieje się w aktualnym czasie, natomiast najczęściej wiąże się ona z zespołem czynników społecznych, literackich i psychologicznych, które mają wpływ na powstanie danego dzieła.

Nieodłącznym elementem występującym w każdej książce są także bohaterowie. Częstość ich występowania, a także definicja głównego bohatera jest prosta do zdefiniowania, chyba, że autor książki jest jej głównym bohaterem bądź narracja jest prowadzona w sposób pierwszoosobowy. W wielu wypadkach wyciągnięcie bohaterów jest proste poprzez wiele kryteriów jakie dana nazwa musi spełniać m.in. duża litera na początku słowa, nie może być znakiem alfanumerycznym, wyraz przypominający nazwisko, częstość występowania danego słowa w tekście, zgromadzenie czasowników w obrębie danego słowa, niekiedy też przymiotników oraz rzeczowniki które w fikcji literackiej mogą być również bohaterami. Język angielski pomaga także w definicji takich słów niekiedy obarczając słowa przedrostkiem „a” lub „the”. Ponadto w przypadku imion i nazwisk, brane pod uwagę będą właśnie nazwiska, gdyż są one w stanie lepiej zidentyfikować danego bohatera, w przypadku podobnych imion. Usuwane są także duplikaty oraz zostały wprowadzone wagi, które zliczają już tak przefiltrowane słowa i końcowo wybierają te najczęściej występujące. Takie kryteria zostały uwzględnione w algorytmach aplikacji Machine Extract Book, a przykładowa ich implementacja przedstawiona na Rys. 5.1.



Rys. 5.1 Przykładowe implementacja algorytmu wyboru bohaterów utworu w aplikacji Machine Extract Book

Ważnym również aspektem danego utworu literackiego jest jego podział. Niekiedy tekst jest podzielony na rozdziały, które są pogrupowane tematycznie, bądź wnoszą nowe informacje dotyczące bohaterów, akcji, czasu bądź budują stopniowo napięcie. Dają także pewną systematykę prowadzenia narracji i wprowadzają czytelnika w stan bezpieczeństwa, poprzez uporządkowaną ilość dostarczanych informacji. W aplikacji Machine Extract Book podział na rozdziały występuje według algorytmu, który sprawdza puste linie (1 lub 2) pomiędzy linijką tekstu. Brane pod uwagę jest także wystąpienie słów kluczowych takich jak „chapter” w takiej linijce, ale także wystąpienie liczby rzymskiej bądź arabskiej na początku lub końcu tekstu. Również taka liczba może się kończyć kropką. Na samym końcu sprawdzana jest poprawna kolejność takich liczb, gdyż mogą być to wartości odbiegające od siebie, dlatego algorytm przy poprawnym uporządkowaniu trzech liczb, jest już w stanie stwierdzić, iż mogą to być potencjalne rozdziały w tekście. Na końcu algorytm wybiera wśród wszystkich takich utworzonych list najliczniejszą i stwierdza, iż mogą być to rozdziały oraz ich pozycje w tekście. Przykładowa implementacja tego algorytmu została przedstawiona na Rys. 5.2.



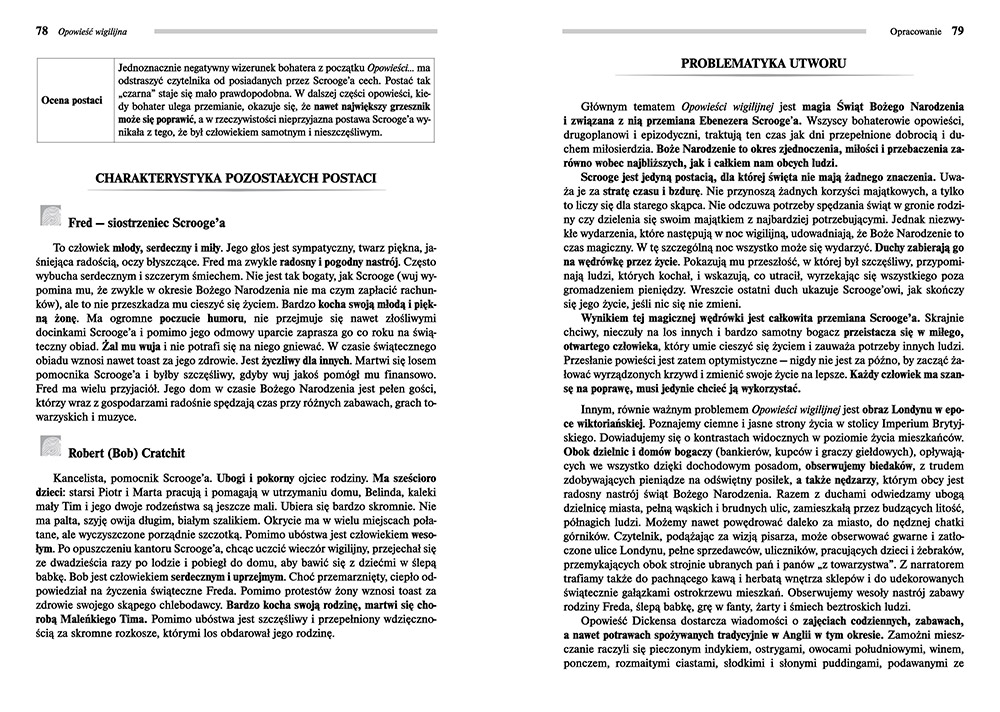
Rys. 5.2 Przykładowe implementacja algorytmu znajdowania rozdziałów w utworze w aplikacji Machine Extract Book

Dialogi występujące w tekście, długie i krótkie niekiedy definiują relacje pomiędzy danymi bohaterami, ale też szybkość czytanego tekstu. Duża liczba dialogów może wskazywać na tekst szybki do przeczytania, a także dużą wymianę zdań pomiędzy bohaterami i dyskusje jakie się w nich toczą. Może też to wnieść informacje o natężeniu dynamiki akcji w danym rozdziale, podobnie jak nagromadzenie czasowników. Te informacje są też przydatne w sposobie zdefiniowania narastania tempa akcji, poprzez zgromadzenie lokalnej statystyki występującej w każdym z rozdziałów.

Duża liczba przymiotników zwalnia tempo czytania danego tekstu, ale jest przydatna w sytuacjach, gdzie występują opisy danych bohaterów lokalnie bądź opisy przyrody a także miejsca rozgrywanej akcji, co może być dobrym krokiem ku wnioskowaniu miejsca akcji w całym utworze.

Na końcu najważniejszą informacją jaką posiada prawie każdy utwór literacki jest problematyka tekstu. Niestety jest to wartość końcowa i niekiedy unikalna, która musi zostać poddana głębszej analizie, poprzez osobę, która zgromadzi wyżej opisane informacje, wraz z dokładnym przestudiowaniem danego dzieła, a także własną refleksją. Rozważenie problematyki danego utworu i jego interpretacja niekiedy może się różnić, w stosunku do osoby przeprowadzającej takie badanie. W końcowym procesie jest w stanie zdefiniować odpowiednio problematykę danego utworu i przesłanie jakie niesie ze sobą dane dzieło.

Każdy utwór posiada też unikalne wartości tak jak np. w „Opowieści Wigilijnej” autorstwa Charlesa Dickensa motywy: motyw ducha, motyw Świąt Bożego Narodzenia, motyw skąpca i motyw przemiany. Bardzo trudne byłoby wyciągnięcie takich wartości dla danego utworu, w szczególności dlatego, iż nie przystają one do wzorca występowania w każdym tekście oraz są zależne od akcji i zrozumienia wydarzeń w danym tekście. Przykładowe opracowanie opisanej wyżej książki zostało przedstawione na Rys. 5.3.



Rys. 5.3 Przykładowe opracowanie lektury „Opowieść Wigilijna” autorstwa Charlesa Dickensa (https://bigimg.taniaksiazka.pl/images/popups/769/36807401465KS,2.jpg)

## Metryki trudności czytanego tekstu

W celu mierzenia statystyk czytelności danych tekstów używa się wszelkiego rodzaju indeksów mierzących trudność czytanego tekstu. Zazwyczaj definiują one dla jakich grup wiekowych jest dany tekst przeznaczony oraz długość czytania danego tekstu. Są dostępne dla wielu języków, natomiast najbardziej popularne istnieją dla języka angielskiego.

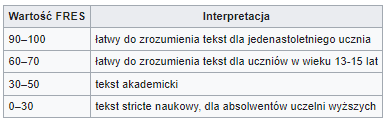
Indeks czytelności Flescha definiuje określenie stopnia w trudności zrozumienia danego tekstu napisanego w języku angielskim. Poziom czytania (The Flesch–Kincaid) został stworzony w ramach kontraktu z marynarką wojenną Stanów Zjednoczonych w 1975 roku przez J. Petera Kincaida, wraz z jego zespołem. Prowadzone badania dotyczyły zaawansowanych technologii, przydatności formuły czytelności, także dla komputerowych tekstów redakcyjnych oraz komputerowego systemu edycji czytelności. Sam indeks został po raz pierwszy użyty przez armię do ocenienia trudności podręczników technicznych w 1978 roku, co wkrótce stało się standardem wojskowym w USA. Pensylwania była pierwszym stanem, który wymagał, aby polisy ubezpieczeniowe komunikacyjne były wystawiane na poziomie trudności czytania nie większym niż dziewiąta klasa, czyli dla osób w wieku 14-15 lat. Jest to też powszechny wymóg w wielu stanach i innych dokumentach prawnych [[29]](#_[Online]_https://pl.wikipedia.org/w_4).

Wartość indeksu jest wyliczana poprzez liczbę słów, zdań w tekście oraz sylab. Im niższa wartość indeksu, tym tekst jest trudniejszy do zrozumienia. Wzór obliczający indeks czytelności został przedstawiony na Rys. 5.4:



Rys. 5.4 Wzór obliczający indeks czytelności Flescha (https://pl.wikipedia.org/wiki/Indeks\_czytelno%C5%9Bci\_Flescha)

Tabela, na której można sprawdzić interpretację wartości indeksu dla tekstów jest przydana w celu rozróżnienia grupy wiekowej, dla jakiej dany utwór jest przeznaczony i została przedstawiona na Rys. 5.5.



Rys. 5.5 Tabelka interpretacji indeksu czytelności Flescha (https://pl.wikipedia.org/wiki/Indeks\_czytelno%C5%9Bci\_Flescha)

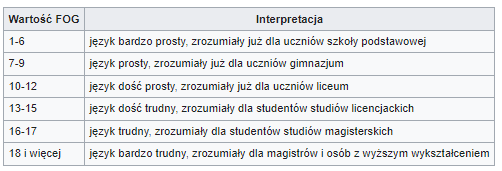
W lingwistyce wskaźnik mgły (Gunning fog index – FOG) jest testem czytelności dla pisania angielskiego. Indeks szacuje lata formalnej edukacji jakie dana osoba musi przeżyć, aby zrozumieć czytany tekst podczas pierwszego czytania. Test został opracowany w 1952 roku przez Roberta Gunninga, amerykańskiego biznesmena, który był zaangażowany w wydawanie gazet i podręczników. Indeks jest powszechnie używany do potwierdzenia, iż tekst trafiający do docelowych odbiorców jest odpowiedni. Teksty wymagające niemal powszechnego zrozumienia są na wartościach poniżej 8 [[30]](#_[Online]_https://pl.wikipedia.org/w_5).

Wartość indeksu jest wyliczana poprzez liczbę słów, zdań w tekście oraz liczbę długich słów (ponad średnią dla danego słowa). Wzór obliczający indeks czytelności został przedstawiony na Rys. 5.6.



Rys. 5.6 Wzór obliczający indeks czytelności FOG (https://pl.wikipedia.org/wiki/Indeks\_czytelno%C5%9Bci\_FOG)

Tabela, na której można sprawdzić interpretację wartości indeksu dla tekstów została przedstawiona na Rys. 5.7.



Rys. 5.7 Tabelka interpretacji indeksu czytelności FOG (https://pl.wikipedia.org/wiki/Indeks\_czytelno%C5%9Bci\_FOG)

Ostatnim wskaźnikiem czytelności tekstu używanym w opisanej pracy i jest indeks SMOG. „Simple Measure of Gobbledygook” w skrócie SMOG jest miarą czytelności tekstu, która szacuje lata nauki potrzebne do zrozumienia fragmentu pisma. Dany indeks jest stosowany do zrozumienia wiadomości o stanie zdrowia wraz z ocenami czytelników, którzy w 100 % zrozumieli materiały testowe. Wzór do obliczenia wartości indeksu SMOG został opracowany przez G. Harry'ego McLaughlina w 1969 roku [[31]](#_[Online]_https://en.wikipedia.org/w_5).

Wartość indeksu jest wyliczana poprzez liczbę zdań i polisylab w tekście, czyli słów, które zawierają co najmniej 3 sylaby. Wzór obliczający indeks czytelności został przedstawiony na Rys. 5.8.



Rys. 5.8 Wzór obliczający indeks czytelności SMOG (https://en.wikipedia.org/wiki/SMOG)

Wyżej opisane wskaźniki są bardzo przydatne do określania czytelności tekstów jak i książek. Potrafią zdefiniować poziom trudności czytanego tekstu jak i jego przeznaczenie dla grupy wiekowej, co może też być pomocne w dobieraniu lektur dla uczniów w szkołach przy analizie książek i ich ilości.

## Porównanie zebranych danych

Po procesie tworzenia aplikacji przystąpiono do wyciagnięcia informacji na podstawie kilku popularnych lektur dostępnych do powszechnego użytku, znajdujących się na stronie: [*https://www.gutenberg.org/*](https://www.gutenberg.org/)*.* Poniżej zostało przedstawione porównanie informacji wyciągniętych z programu, wraz z wnioskami jakie za sobą niosą oraz z informacjami zawartymi w streszczeniach szczegółowych. Podział analizy został przedstawiony następująco, na początku zostaną omówione ogólne informacje jakie niesie za sobą dany utwór literacki, a następnie przedstawione zostaną statystyki rozdziałowe wraz z zawartymi w nich informacjami.

Pierwszą książką poddaną analizie została „Opowieść wigilijna” Charlesa Dickensa. Książka opowiada o samotnym bohaterze imieniem Scrooge, który ze swojej natury jest skąpcem i pragnie jak to każdego roku spędzać święta Bożego Narodzenia samemu, gdyż uważa to za święto ludzi, którzy nie potrafią niczego osiągnąć w życiu. Tego rodzaju święta nigdy nie traktował zbyt poważnie.

Analiza danej książki zawiera podstawowe wartości definiujące długość książki w znakach i słowach, liczba zdań, oraz ich statystyki w podobnych kryteriach. Ponadto występują też wartości definiujące ilości dialogów oraz statystyki z nimi związane. Do dyspozycji są również określenia czasu akcji zawarte w książce oraz informacje o częstości występowania bohaterów w tekście. Dane podstawowe dla pierwszej opisywanej książki zostały zebrane w tabelce Tab. 5.3.1.

|  |  |
| --- | --- |
| *Nazwa* | *Wartość* |
| Długość książki w znakach | 164530 |
| Długość książki w słowach | 26550 |

*Tab. 5.3.1 Podstawowe statystyki wyciągnięte z książki „Opowieść wigilijna” Charlesa Dickensa*

Z podstawowych informacji zawartych w książce można zauważyć, że książka ma około 27 tys. słów i ponad 160 tys. znaków. Można wywnioskować, że czas jej przeczytania to od kilku do kilkunastu godzin. Książka zawiera około 96 stron [[36]](#_[Online]_https://www.gutenberg.org/). Przy czytaniu 10 stron na godzinę, według kalkulatora szybkości czytania zawartego na stronie [*https://www.omnicalculator.com/everyday-life/reading-time*](https://www.omnicalculator.com/everyday-life/reading-time), można oszacować, że czas przeczytania „Opowieści wigilijnej” to 9 godzin i 35 minut. Następnym ważnym aspektem jest statystyka dialogowa. Jej wartości zostały przedstawione w tabelce Tab. 5.3.2.

|  |  |
| --- | --- |
| *Nazwa* | *Wartość* |
| Liczba dialogów | 682 |
| Średnia liczba słów w dialogach | 17 |
| Średnia liczba znaków w dialogach | 71 |
| Liczba słów w długich dialogach | 148 |
| Liczba słów w krótkich dialogach | 534 |

*Tab. 5.3.2 Podstawowe statystyki dialogowe wyciągnięte z książki „Opowieść wigilijna” Charlesa Dickensa*

Ogólna liczba dialogów w książce to koło 700. Średnia liczba słów w dialogu to 17, a znaków to 71. Tekst jest w miarę dynamiczny, dzięki czemu szybko się go czyta. Często występują też dialogi między różnymi osobami np. głównym bohaterem i duchami zawarte w tekście. W tekście dominują krótkie dialogi, zatem można wywnioskować, że jest tam zawarta szybka wymiana zdań między bohaterami. Ogólny wykres ilości dialogów w książce został zawarty na wykresie Wyk. 5.3.1.

*Wyk. 5.3.1 Statystyka dialogowa dla książki „Opowieść wigilijna” Charlesa Dickensa*

Ostatnim ważnym aspektem jest stosunek czasów zawartych w książce. Definiuje on czas akcji zawarty w książce. Ponadto można wyciągnąć wnioski w jakim stosunku tekst był opisowy i jak zmieniał się czas akcji. Ogólna statystyka podziału czasów została przedstawiona w tabelce Tab. 5.3.3 i wykresie Wyk. 5.3.2. Pokazuje ona m. in dominację czasu przeszłego nad teraźniejszym. Oznacza to, że w książce mamy wiele razy odniesienie do retrospekcji lub wydarzeń występujących przed rozpoczęciem głównego wątku akcji. Jest to zgodne, gdyż w książce wiele razy jest odwołanie do osób bądź miejsc występujących w przeszłości.

|  |  |
| --- | --- |
| *Nazwa* | *Wartość* |
| Liczba czasowników | 3485 |
| Liczba czasowników w czasie teraźniejszym | 1115 |
| Liczba czasowników w czasie przeszłym | 2370 |

*Tab. 5.3.3 Statystyki czasowe wyciągnięte z książki „Opowieść wigilijna” Charlesa Dickensa*

*Wyk. 5.3.2 Statystyki czasów w wykresie kołowym dla książki „Opowieść wigilijna” Charlesa Dickensa*

Ponadto program opisany w następnych rozdziałach jest w stanie wyciągnąć bohaterów z danej książki na podstawie m.in częstotliwości ich występowania w książce w stosunku do ilości słów. Wynik kilkuprocentowy może świadczyć o głównych bohaterach tekstu. Program jest w stanie wyciągnąć dowolną liczbę głównych bohaterów w zależności od zdefiniowania zakresu dla algorytmu użytego w środku. Dany użytkownik musi sam ocenić poprawność wyciągniętych informacji. Natomiast w wielu przypadkach wśród ilości propozycji znajdują się główni bohaterowie książki. W poniższej tabelce zostali przedstawieni najważniejsi bohaterowie dla opisywanej książki. Zostali wyciągnięci po odrzuceniu wartości mylących i przedstawieni w tabelce Tab. 5.3.4 i są to główni bohaterowie w książce.

|  |
| --- |
| *Nazwa* |
| Scrooge |
| Marley |
| Fezziwig |

*Tab. 5.3.4 Główni bohaterowie książki „Opowieść wigilijna” Charlesa Dickensa*

Prócz statystyk globalnych dla danej książki można wyciągnąć statystyki lokalne, jeżeli dana książka dzieli się na rozdziały. Rozważana książka dzieli się na pięć rozdziałów, które zostaną przeanalizowane według wyciągniętych informacji zawartych w streszczeniu szczegółowym. Informacje zawarte o statystykach rozdziałowych zostały zawarte w tabeli Tab. 5.3.5.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Liczba słów** | **Liczba znaków** | **Liczba słów w zdaniu** | **Liczba znaków w zdaniu** | **Liczba zdań** | **Liczba dialogów** |
| **1** | 5906 | 36340 | 13,69 | 73,64 | 446 | 137 |
| **2** | 5697 | 34239 | 13,83 | 72,18 | 421 | 120 |
| **3** | 7593 | 46364 | 17,26 | 93,51 | 447 | 110 |
| **4** | 4912 | 28919 | 11,96 | 61,4 | 435 | 134 |
| **5** | 2206 | 13120 | 9,49 | 47,03 | 248 | 66 |
|  |  |  |  |  |  |  |
|  | **Liczba długich dialogów** | **Liczba krótkich dialogów** | **Czasowniki ogólnie** | **Czasowniki w czasie przeszłym** | **Czasowniki w czasie teraźniejszym** | **Liczba przymiotników** |
| **1** | 19 | 118 | 768 | 444 | 324 | 384 |
| **2** | 27 | 93 | 723 | 455 | 268 | 387 |
| **3** | 22 | 88 | 944 | 598 | 346 | 560 |
| **4** | 29 | 105 | 709 | 416 | 293 | 310 |
| **5** | 17 | 49 | 292 | 158 | 134 | 112 |

*Tab. 5.3.5 Statystyki lokalne dla rozdziałów wyciągnięte z książki „Opowieść wigilijna” Karol Dickensa. Kolory w kolumnach odpowiadają kolejno: zielony – największa wartość, czerwony – najgorsza wartość*

Liczba słów i znaków rośnie i jest kulminacyjna w trzecim rozdziale. Najprawdopodobniej jest to najważniejszy rozdział, gdzie występuje kulminacja akcji w książce. Co za tym idzie, można rozważyć, iż Charles Dickens lub budować napięcie w swoich książkach. Ponadto liczba słów i znaków w zdaniach najgorzej wypada w ostatnich rozdziałach. Może to być spowodowane tym, iż jest tam epilog książki lub napięcie i akcja nie są znaczące. Co więcej najwięcej dialogów występuje w czwartym rozdziale. Jest tam przedstawiona rozmowa Scrooge’a razem z Duchem Przyszłych Świąt. Kupcy rozmawiają tam bardzo dużo o pieniądzach, które liczą się dla głównego bohatera przede wszystkim. Najkrótsze dialogi występują na początku książki, gdzie następuje opis głównego bohatera, jego byłego wspólnika Jakuba Marleya oraz krótkie rozmowy razem z innymi bohaterami. W książce dominuje także czas teraźniejszy nad przeszłym. W pierwszym rozdziale znajduje się opis i wstęp do książki i różnice czasowe są tam nieznaczne. Natomiast w drugim i trzeci rozdziale, gdzie rozpoczyna się akcja utworu i rozmowa z duchami widać przewagę czasu teraźniejszego nad przeszłym [[36]](#_https://www.gutenberg.org/files/46/).

Ostatnim ważnym czynnikiem jest trudność czytanej książki. W poprzednim rozdziale zostały opisane trzy zmienne definiujące trudność czytanego tekstu. Dla „Opowieści wigilijnej” indeks czytelności Flescha ma następującą wartość:

Co oznacza, że tekst jest łatwy do przeczytania dla uczniów w wieku 13-15 lat. ,,Opowieść wigilijna” znajduje się w liście lektur dla klasy VII, więc wartość wskaźnika jest zgodna z założeniami. Dana książka dla prostej miary Gobbledygooka (SMOG) otrzymuje wartość:

Dana wartość oznacza, że tekst można przeczytać po około 5 latach nauki w szkole. Lektura jest dostępna dla klasy siódmej, czyli trzynastego roku życia. Więc wartość wskaźnika mieści się w wartości błędu. Ostatni wskaźnik formuły Gunning fog, która również określa liczbę lat, po których tekst będzie zrozumiały, ale bierze pod uwagę inne wartości. Dla „Opowieści wigilijnej” ma on wartość:

Połowa tej wartości jest równa 4.66 i tyle wynosi liczba lat po której przeczytanie danej książki nie powinno sprawić problemu i jest zbliżona do wartości współczynnika SMOG, zatem oba współczynniki są zgodne.

Drugą książką poddaną analizie została książka pod tytułem „The History of Freedom and Other Essays” autorstwa Johna Emericha Edwarda Dalberga-Actona. Książka stanowi doskonałe wprowadzenie w myślenie tego wybitnego myśliciela, uważanego za jednego z najbardziej uczonych Anglików swoich czasów, który stworzył historię wolności jako dzieło swojego życia. Rzeczywiście uważał wolność polityczną za niezbędny warunek i strażnika wolności religijnej. Jest to utwór z gatunków historycznych opowiadający też o wojnach napoleońskich w Europie i skupiająca się na wydarzeniach historycznych [[37]](#_https://www.gutenberg.org/ebooks/31).

W podobny sposób pierwszym sprawdzanym kryterium będzie długość książki w znakach i słowach, liczba zdań, dialogów oraz statystyki z nimi związane. Do dyspozycji jest również czas zawarty w książce oraz informacje o częstości występowania bohaterów w tekście, definiujące bohaterów i głównego bohatera. Podstawowe statystyki czasowe zostały zebrane w tabelce Tab. 5.3.6 i Wyk. 5.3.3.

|  |  |
| --- | --- |
| *Nazwa* | *Wartość* |
| Liczba czasowników | 32567 |
| Liczba czasowników w czasie teraźniejszym | 14275 |
| Liczba czasowników w czasie przeszłym | 18292 |

*Tab. 5.3.6 Podstawowe statystyki czasowe wyciągnięte z książki „The History of Freedom and Other Essays” autorstwa John Emerich Edward Dalberg-Acton*

*Wyk. 5.3.3 Statystyki czasów w wykresie kołowym dla książki „The History of Freedom and Other Essays” autorstwa John Emerich Edward Dalberg-Acton*

Uzyskane wartości informują, że wydarzenia dzieją się w czasie przeszłym jak i teraźniejszym w podobnym stosunku wartości. Wynika to z tego, iż autor przedstawiając wydarzenia historyczne obarcza je komentarzem i własnym tokiem myślenia, tak jak zostało to uwzględnione w opisie książki.

Podobnie program może wyciągnąć bohaterów z danej książki oraz częstotliwość ich występowania w książce w stosunku do ilości słów. Program jest w stanie wyciągnąć dowolną liczbę głównych bohaterów w zależności od kryteriów. Dany użytkownik musi sam ocenić poprawność wyciągniętych informacji. Natomiast w wielu przypadkach wśród ilości propozycji znajdują się główni bohaterowie książki. W poniższej tabelce zostali przedstawieni najważniejsi bohaterowie, którzy występowali w książce. Książka posiadała długą treść, także o miejscach akcji i nazw rozgrywanych bitew, które autor czasem traktuje w sposób osobowy, dlatego należało ręcznie odrzucić wartości błędne. Wyciągnięci bohaterowie po odrzuceniu takich wartości zostali przedstawieni w tabelce Tab. 5.3.7 i są to główni bohaterowie w książce:

|  |
| --- |
| *Nazwa* |
| Napoleon |
| Eric |
| Mackreth |

*Tab. 5.3.7 Główni bohaterowie książki „The History of Freedom and Other Essays” autorstwa John Emerich Edward Dalberg-Acton*

Prócz statystyk globalnych w podobny sposób dla obecnej książki można wyciągnąć statystyki lokalne, jeżeli dana książka dzieli się na rozdziały. Rozważana książka dzieli się na kilkanaście rozdziałów. Informacje te zostały zawarte w tabeli Tab. 5.3.8.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Liczba słów** | **Liczba znaków** | **Liczba słów w zdaniu** | **Liczba znaków w zdaniu** | **Liczba zdań** | **Liczba dialogów** |
| **1** | 10196 | 60202 | 358 | 166,09 | 28,41 | 2 |
| **2** | 10820 | 63765 | 383 | 164,53 | 28,29 | 11 |
| **3** | 13667 | 82590 | 559 | 145,75 | 24,45 | 2 |
| **4** | 20802 | 122527 | 1404 | 85,38 | 14,97 | 32 |
| **5** | 16341 | 100643 | 832 | 118,86 | 19,73 | 172 |
| **6** | 8623 | 51546 | 400 | 126,8 | 21,6 | 52 |
| **7** | 6799 | 41618 | 250 | 164,89 | 27,6 | 117 |
| **8** | 13553 | 80506 | 550 | 144,38 | 24,66 | 37 |
| **9** | 10809 | 65634 | 389 | 166,7 | 27,82 | 24 |
| **10** | 28147 | 167543 | 964 | 171,81 | 29,24 | 52 |
| **11** | 21614 | 128047 | 802 | 157,7 | 26,99 | 49 |
| **12** | 8894 | 52766 | 271 | 192,74 | 32,85 | 15 |
| **13** | 11023 | 65403 | 437 | 147,71 | 25,27 | 9 |
| **14** | 22025 | 135130 | 877 | 152,05 | 25,16 | 27 |
| **15** | 8437 | 50474 | 383 | 129,84 | 22,11 | 49 |
| **16** | 4576 | 26944 | 187 | 142,04 | 24,66 | 41 |
| **17** | 24045 | 149440 | 678 | 218,74 | 35,79 | 32 |
|  |  |  |  |  |  |  |
|  | **Liczba długich dialogów** | **Liczba krótkich dialogów** | **Czasowniki ogólnie** | **Czasowniki w czasie przeszłym** | **Czasowniki w czasie teraźniejszym** | **Liczba przymiotników** |
| **1** | 1 | 1 | 1690 | 870 | 820 | 1087 |
| **2** | 4 | 7 | 1795 | 806 | 989 | 1069 |
| **3** | 1 | 1 | 2292 | 1084 | 1208 | 1435 |
| **4** | 11 | 21 | 3967 | 2071 | 1896 | 2195 |
| **5** | 75 | 97 | 2740 | 1719 | 1021 | 1975 |
| **6** | 16 | 36 | 1353 | 877 | 476 | 994 |
| **7** | 40 | 77 | 1452 | 1006 | 446 | 1013 |
| **8** | 8 | 29 | 2123 | 1423 | 700 | 1386 |
| **9** | 7 | 17 | 1671 | 983 | 688 | 1280 |
| **10** | 20 | 32 | 4386 | 2683 | 1703 | 2791 |
| **11** | 21 | 28 | 3774 | 1745 | 2029 | 2302 |
| **12** | 6 | 9 | 1545 | 1065 | 480 | 889 |
| **13** | 3 | 6 | 1885 | 1207 | 678 | 1054 |
| **14** | 7 | 20 | 4316 | 1942 | 2374 | 2390 |
| **15** | 17 | 32 | 1393 | 893 | 500 | 848 |
| **16** | 11 | 30 | 862 | 587 | 275 | 534 |
| **17** | 9 | 23 | 2037 | 1285 | 752 | 1570 |

*Tab. 5.3.8 Statystyki lokalne dla rozdziałów wyciągnięte z książki „The History of Freedom and Other Essays” autorstwa John Emerich Edward Dalberg-Acton. Kolory w kolumnach odpowiadają kolejno: zielony – największa wartość, czerwony – najgorsza wartość*

Liczba słów i znaków rośnie jest zróżnicowana względem rozdziałów, co oznacza, że tekst ma bardziej charakter opisowy niż narastające tempo i kulminację akcji. Najprawdopodobniej cała książka opisuje wydarzenia dziejące się w przeszłości z komentarzem autora i jego refleksjami. Duża liczba czasowników jest zgromadzona w rozdziale 10, możliwe, iż występuje tam bitwa jaką przeprowadzał Napoleon. Jest też tam najwięcej przymiotników, co sugeruje, że scena prócz swojej dynamiki mogła mieć też charakter opisowy – pola bitwy. Zdania, słowa w zdaniu i znaki są także zróżnicowane, co oznacza, iż problematyka jest po równo rozważana w tekście nie kładąc nacisku na żaden z rozdziałów. Ogólna liczba znaków, słów i przymiotników jest bardzo duża, więc daną książkę należy przeczytać ze zrozumieniem w dłuższym czasie niż normalną powieść fabularną. Ponadto mała liczba dialogów również sugeruje, iż książka nie jest za bardzo dynamiczna, a dialogów między bohaterami w zasadzie nie ma. Nie występują burzliwe konwersacje, które mogłyby wpłynąć na akcję danego utworu lub wydarzeń w nim.

Ostatnim również ważnym czynnikiem jest trudność czytanej książki. Dla książki „The History of Freedom and Other Essays” indeks czytelności Flescha ma następującą wartość:

Oznacza to, iż tekst jest już trudny i może być do rozważany przez studentów, w trakcie analizy danego dzieła. Książka porusza przemyślenia autora na tematy historyczne i faktycznie może nie nadawać się dla osób będących w wieku wczesno szkolnym, więc wartość wskaźnika jest zgodna z założeniami. Dana książka dla prostej miary Gobbledygooka (SMOG) otrzymuje wartość:

Dana wartość oznacza, że tekst można przeczytać po około 7 latach nauki w szkole. Może to również odpowiadać wczesnym latom akademickim dla danego ucznia. Ostatni wskaźnik formuły Gunning fog również określa ilość lat, po których tekst będzie zrozumiały, ale bierze pod uwagę inne wartości. Dla danej książki ma on wartość:

Co zgodnie z założeniami tabeli wyżej przedstawionej, oznacza, iż jest to tekst trudny, przeznaczony dla studiów magisterskich do analizy danego dzieła. Co również jest zgodne z założeniami.

Powyższe wyniki przedstawiają, iż program dla dwóch skrajnie różnych gatunków literackich oraz różnej tematyce obu utworów wyciągnął poprawne informacje i uprościł proces wstępnej analizy danego dzieła do minimum, przyśpieszając go również. Dzięki zaoszczędzonemu czasu, czytelnik wie już, że pierwszy utwór jest w stanie szybko przeczytać i zrozumieć problematykę w nim zawartą, natomiast do drugiego musi się przygotować i przestudiowanie go zajmie mu zdecydowanie więcej czasu, aby zrozumieć przesłanie przekazywane przez autora.

## Miejsce i czas akcji

Podczas tworzenia aplikacji została podjęta próba wyciągnięcia informacji lokalizacyjnych w rozdziale, aby zdefiniować prawdopodobne miejsce akcji lub miejsca w których odbywała się dana akcja w każdym z rozdziałów. Niestety próba zakończyła się niepowodzeniem, gdyż jakkolwiek w przypadku miejsc historycznych dane dawały dobre rezultaty i mimo wartości szumów użytkownik był w stanie zdefiniować prawdopodobną lokalizację, to w przypadku fikcji literackiej, takie wartości były bardzo mylne i często zazębiały się z informacjami o bohaterach. Przykładowe informacje wyciągnięte z omawianej wcześniej „Opowieści Wigilijnej” zostały przedstawione w Tab. 5.3.9.

|  |
| --- |
| Spectre |
| Ignorance |
| Poulterer |
| Cold Roast |
| Belshazzars |
| Camden Town |
| Truth |
| Abrahams |
| Turkey |
| Great Britain |
| Saint Dunstan |
| Belle |
| United States' |
| the City of London |
| London |
| Plenty |

*Tab. 5.3.9 Przykładowe informacje lokalizacyjne wyciągnięte z książki „Opowieść wigilijna” Charlesa Dickensa*

Wskazuje to, iż pomimo znalezienia prawidłowej lokalizacji: Londyn, gdzie faktycznie toczy się akcja danej książki, zostały wyciągnięte inne kraje sugerujące narodowości (Turcja) niektórych bohaterów książki. Precyzja 0,2 jest niezadowalająca i bardziej może wprowadzić czytelnika w błąd niż przynieść korzystne efekty przy analizie danego dzieła literackiego.

Podobna sytuacja występuje w przypadku wyciągnięcia czasu akcji dla danego wydarzenia. Z wyciągniętych słów można określić, iż akcja występuje w okresie zimowym, w Boże Narodzenie. Natomiast duża liczba słów pobocznych jest trochę myląca, gdyż zawiera w sobie określenia dni, tygodni i miesięcy. Przykładowe wartości zostały przedstawione w Tab. 5.3.10.

|  |
| --- |
| Christmas Day |
| their holiday |
| the four quarters |
| winter day |
| 42 |
| Christmas |
| spring |
| nearly seventy years |
| Christmas Eve old |
| the very day |
| the previous Monday |
| day |
| no days |
| this festive season of the year |
| Saturdays |
| the wild year |
| the year |
| seven years |
| The quarter |
| these seven years |
| three days |
| years ago |
| the day before |
| these days |
| this day |
| three quarters |
| Sunday |
| weekly |
| Christmas Eve |
| a-week |
| week |
| all the year |
| every year |

*Tab. 5.3.10 Przykładowe informacje czasowe wyciągnięte z książki „Opowieść wigilijna” Charlesa Dickensa*

Wskazuje to prawdopodobnie na retrospekcje związane z przenoszeniem się pomiędzy świętami lub retrospekcjami zawartymi w „Opowieści Wigilijnej”, natomiast może być mylące dla użytkownika. Może stwierdzić on, iż akcja dzieje się bardzo dynamicznie w wielu obrębach czasowych, gdyż tak naprawdę akcja dzieje się w jedne święta i doprowadzi go to do mylnych wniosków. Wyciąganie dat i informacji o czasie z biblioteki spaCy jest bardzo kuszące, aczkolwiek algorytmy nie są jeszcze na tyle zaawansowane, aby odrzucać informacje przydatne dla użytkownika oraz ograniczyć obszar poszukiwań do niezbędnego minimum. Ponadto w wielu książkach czas nie jest bezpośrednio zdefiniowany. Czas i miejsce akcji wielokrotnie wynika z genezy powstania książki lub opisu jaki jest w niej zawarty. Informacje o nim występują nawet i w problematyce utworu. Przy obecnym poziomie technologicznym nawet najskuteczniejsze do tej pory algorytmy nie będą w stanie dokładnie wyciągnąć takich informacji.

# Projekt aplikacji do ekstrakcji informacji z dzieła literackiego

## Założenia aplikacji

Tworzona w ramach opisywanej pracy aplikacja ma udostępniać użytkownikowi następujące funkcjonalności:

* Możliwość załadowania danego dzieła literackiego lub tekstu w pliku z rozszerzeniem .txt
* Możliwość wyciągnięcia statystyk globalnych dla danego tekstu, w tym:
* Liczba znaków
* Liczba słów
* Liczba zdań
* Średnia liczba znaków w zdaniach
* Średnia liczba słów w zdaniach
* Liczba czasowników
* Liczba czasowników w czasie teraźniejszym
* Liczba czasowników w czasie przeszłym
* Stosunek procentowy czasowników w czasie teraźniejszym do czasu przeszłego
* Współczynnik czytelności FRE
* Współczynnik czytelności FOG
* Współczynnik czytelności SMOG
* Liczba przymiotników
* Liczba dialogów
* Średnia liczba znaków w dialogach
* Średnia liczba słów w dialogach
* Liczba długich dialogów
* Liczba krótkich dialogów
* Stosunek procentowy długich dialogów do krótkich
* Bohaterowie przykładowi
* Czas wykonania analizy
* Możliwość wyciągnięcia statystyk lokalnych (rozdziałowych) dla danego tekstu w tym:
* Liczba słów
* Liczba znaków
* Liczba zdań
* Średnia liczba znaków w zdaniach
* Średnia liczba słów w zdaniach
* Liczba dialogów
* Liczba krótkich dialogów
* Liczba długich dialogów
* Liczba czasowników
* Liczba czasowników w czasie teraźniejszym
* Liczba czasowników w czasie przeszłym
* Liczba przymiotników
* Bohaterowie
* Widok statystyk czasowych i dialogowych na wykresach
* Możliwość zapisania statystyk lokalnych w pliku z rozszerzeniem .txt i globalnych w pliku .csv do dalszej analizy danych
* Możliwość rozszerzenia funkcjonalności danej aplikacji
* Interfejs graficzny do obsługi aplikacji

Aplikacja ma być w stanie udostępnić wiele funkcjonalności, aby użytkownik mógł być w stanie przeprowadzić wstępną analizę danego dzieła literackiego i skrócić etap jej analizy. Aplikacja może być przydatna w kompletowaniu streszczenia dzieła bądź przygotowywaniu jego opracowania. Program daje możliwość załadowania plików tekstowych z odpowiednim rozszerzeniem oraz wyeksportowania danych do innych plików w celu przeprowadzenia dalszych analiz. Aplikacja może wczytywać pliki tekstowe w lokalizacji podanej przez użytkownika. Jeśli w wyniku analizy nie powstaną błędy, proces zakończy się pomyślnie i udostępni użytkownikowi dane, w przeciwnym razie użytkownik zostanie poinformowany o błędach, np. o złym rozszerzeniu książki. Ponadto nie każdą książkę da się podzielić na rozdziały, o czym program dodatkowo poinformuje użytkownika.

## Wymagania aplikacji

Aktorzy występujący w aplikacji:

**Użytkownik** – korzystający z aplikacji może załadować dany plik tekstowy, przeprowadzić analizę jego zawartości i wyeksportować wyniki

**Program** – przetwarza informacje od użytkownika, poddaje analizie dany plik tekstowy, wyświetla uzyskane podczas niej informacje. Daje możliwość podzielenia pliku na rozdziały, jeśli takie istnieją. Wyciąga statystyki dla rozdziałów i pozwala na zapisywanie ich w celu dalszych operacji.

**Wymagania użytkownika:**

1. Użytkownik powinien mieć możliwość wczytania pliku zawierającego tekst.
2. Użytkownik może zapisać plik z analizą globalną.
3. Użytkownik może zapisać plik z analizą lokalną.
4. Użytkownik powinien móc wyświetlać wykresy w programie.
5. Użytkownik powinien posługiwać się graficznym interfejsem do wprowadzenia zmian.

**Wymagania systemowe:**

1. System powinien przetwarzać plik pobrany od użytkownika i sprawdzać jego poprawność.
2. System w przypadku podania prawidłowego pliku powinien poddać go analizie globalnej i lokalnej.
3. System powinien w przypadku źle wprowadzonego pliku poinformować użytkownika o błędzie.
4. System w przypadku nie wyciągnięcia informacji lokalnych powinien poinformować, iż dana książka może nie posiadać rozdziałów.
5. System ma być dostępny na platformy Windows z zainstalowanym interpreterem języka Python 3.7.6 w wersji 64-bitowej oraz bibliotekami pyqt, spacy oraz pandas.
6. System powinien posiadać przejrzysty interfejs pokazujący aktualny stan aplikacji i oferować różne funkcjonalności dla użytkownika.

**Wymagania funkcjonalne:**

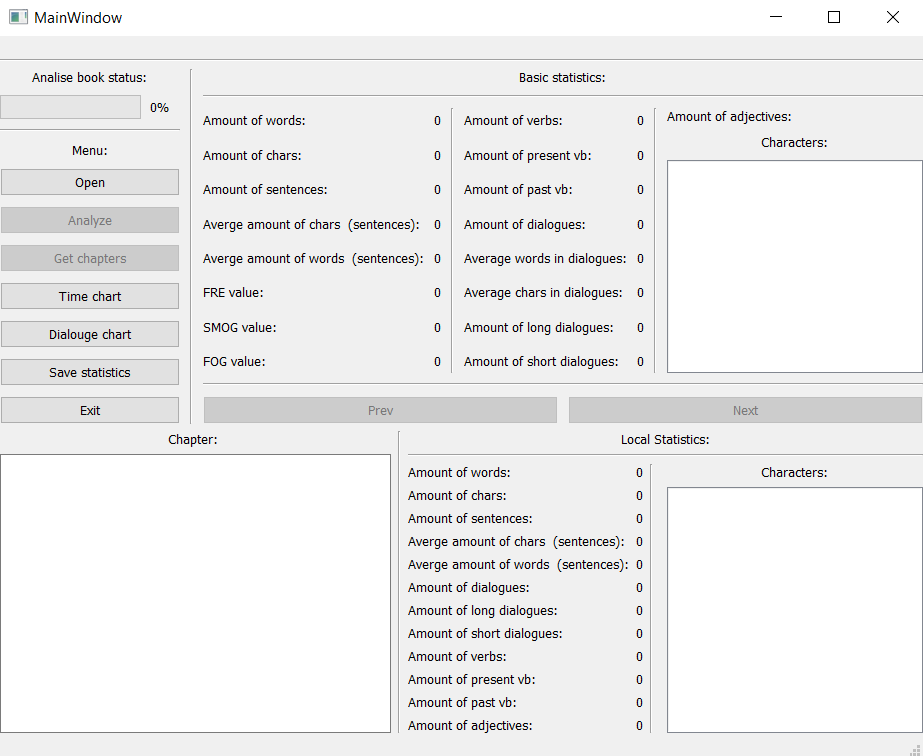
1. Użytkownik ma możliwość korzystania z aplikacji bez rejestracji.
2. Dane użytkownika nie są gromadzone w aplikacji.
3. Rezultat może być zapisywany, jak i również współdzielony między innymi użytkownikami.
4. Możliwość wymiany plików tekstowych między użytkownikami.
5. Użytkownik może edytować pliki w formacie txt za pomocą innego edytora i potem wczytywać je do programu.
6. Osoba korzystająca z aplikacji może tworzyć pliki w formacie txt i csv i zapisywać je, w celu dalszej edycji.

**Wymagania niefunkcjonalne:**

1. Możliwość korzystania z aplikacji jest całodobowa.
2. Nieograniczona liczba przetwarzanych plików przez aplikację.
3. Zmiany nie mogą być wykonywane w plikach w trakcie ich analizy.
4. W trakcie analizy może być przetwarzany tylko jeden plik.
5. W trakcie analizy dostępne są funkcje poglądowe.
6. Operacje użytkownika w programie w trakcie analizy danego pliku są niemożliwe.
7. Zmiany w innych obiektach oddziałują na siebie.
8. Nie jest wymagane połączenie z Internetem lub inną platformą.

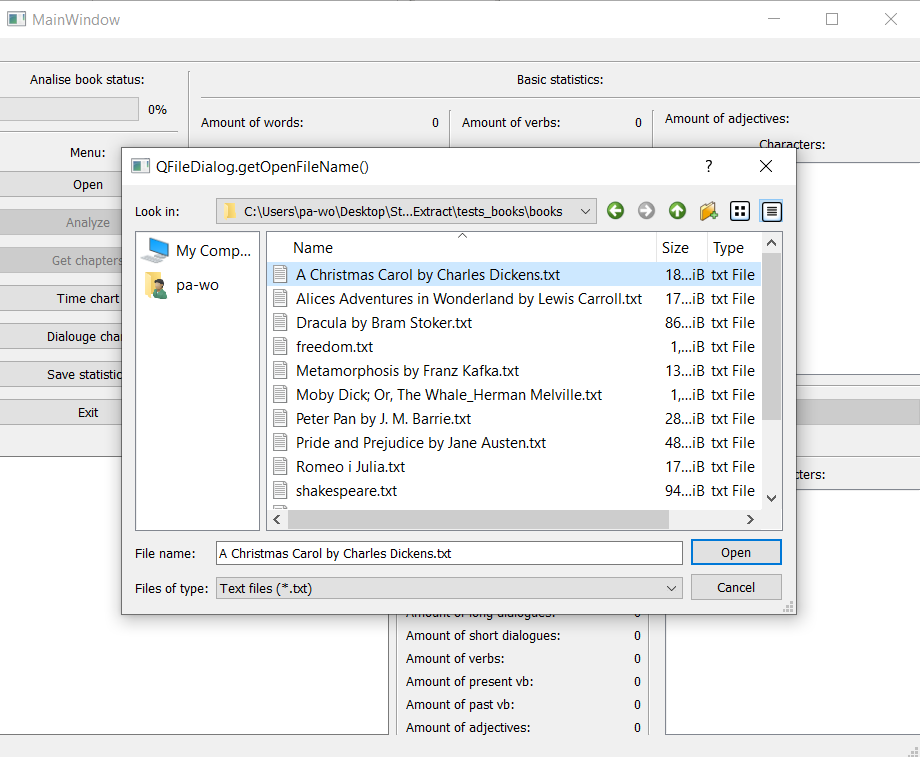
## Obsługa aplikacji

Po zainstalowaniu potrzebnych bibliotek wraz z językiem Python 3.7.6 oraz dodatkowych bibliotek spaCy, coverage, syllables aplikacja jest gotowa do użycia. Po uruchomieniu pliku wykonywalnego pojawia się okno główne *Main Window*, widoczne na Rys. 6.1. witając użytkownika i umożliwiając wyjście z aplikacji przyciskiem *Exit* albo przejście do okna wyboru wczytania danego pliku tekstowego, używając przycisku *Open*.



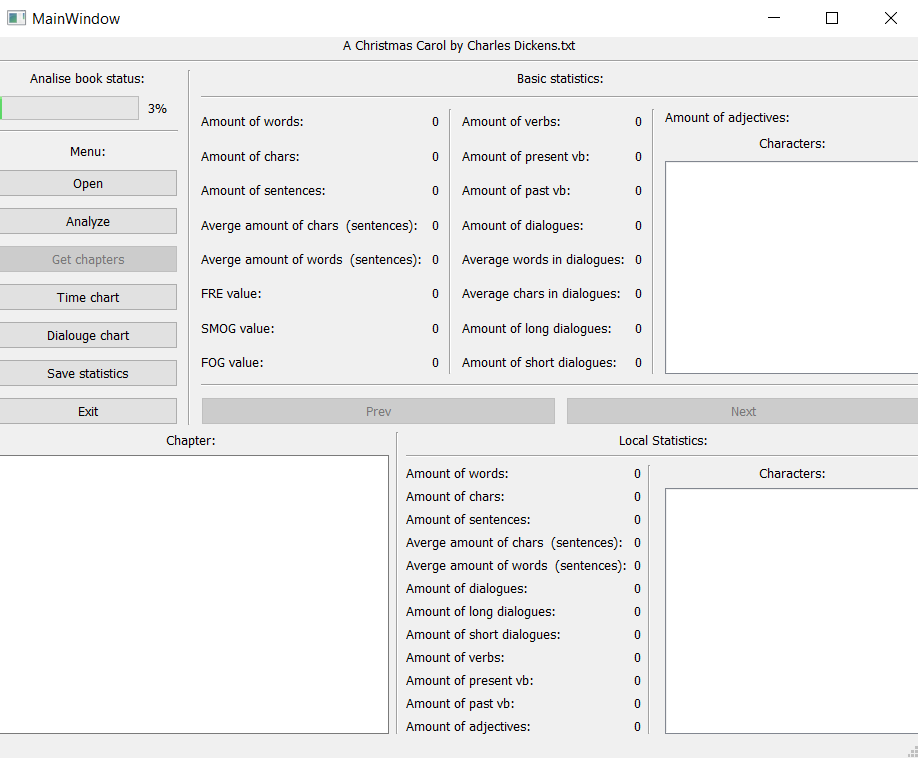
Rys. 6.1 Główne okno programu Machine Extract Book

W przypadku naciśnięcia *Time chart* albo *Dialogue* w głównym oknie pojawi się informacja o braku przeanalizowanego pliku, gdyż taki nie został wczytany ani przetworzony. Podobnie w przypadku naciśnięcia przycisku *Save statistics*, pojawi się odpowiednia informacja informująca użytkownika o braku wykonanych czynności rozpoczynających analizę. Po naciśnięciu przycisku *Open* pojawi się okienko wyboru pliku tekstowego widoczne na Rys. 6.2.



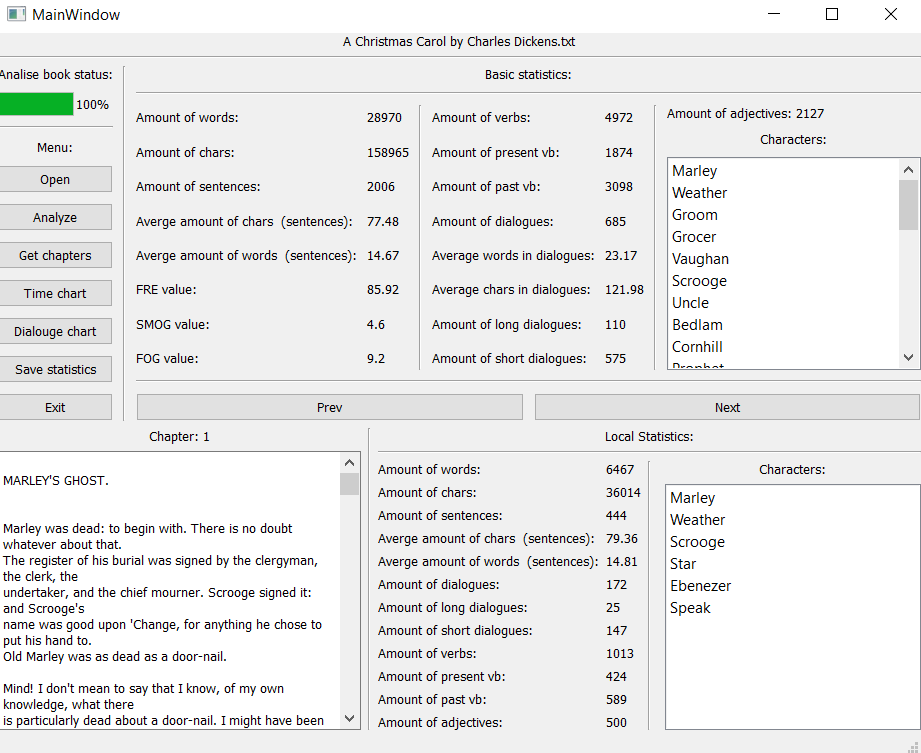
Rys. 6.2 Okno otwierania pliku do przetwarzania w programie Machine Extract Book

Po wybraniu pliku należy nacisnąć przycisk *Open* i plik będzie już gotowy do analizy. Po naciśnięciu przycisku *Analyze* rozpocznie się przetwarzanie danego pliku, które może potrwać zauważalny okres czasu, np. dla książek zawierających 200 000 znaków do 10 sekund i dla książek ponad 1 500 000 znaków ponad 3 minuty. O statusie przetwarzania pliku będzie informował pasek postępu znajdujący się w lewym górnym rogu okienka. Przykładowa analiza pliku została przedstawiona na Rys. 6.3.



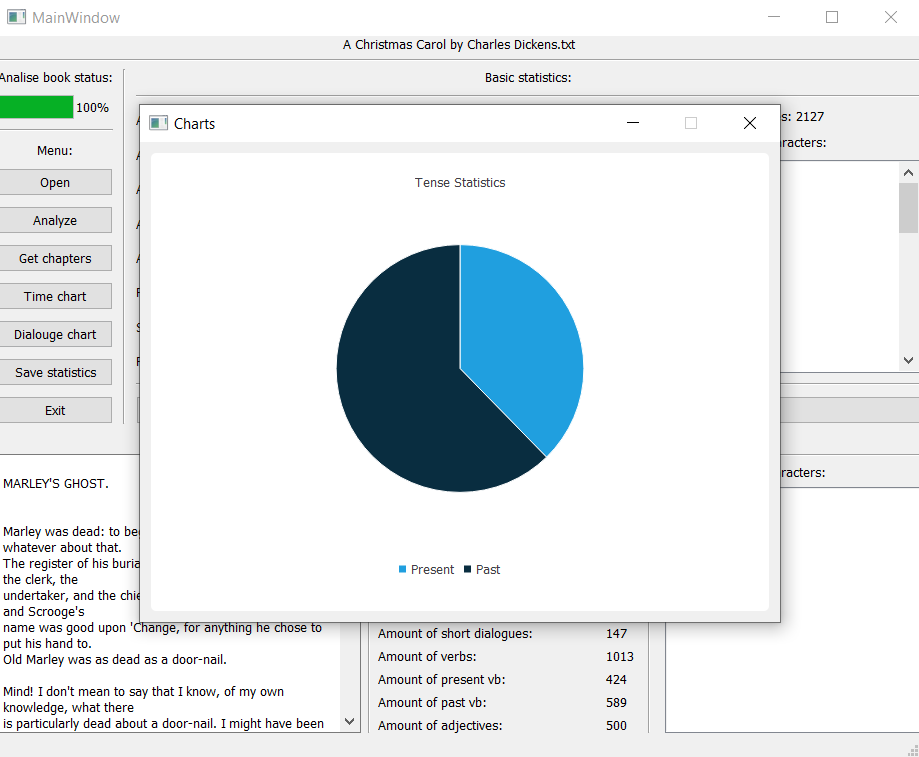
Rys. 6.3 Proces przykładowej analizy pliku tekstowego zawierającego treść książki „Opowieść Wigilijna” Charlesa Dickensa w programie Machine Extract Book

Po przeanalizowaniu danego pliku pojawi się wyciągnięta statystyka globalna dla danego pliku. Jeśli użytkownik zechce dokonać analizy lokalnej, musi wybrać przycisk *Get chapters* i pokaże się statystyka rozdziałowa jakie zostały wyciągnięte z książki, chyba iż takie rozdziały nie istnieją lub nie zostaną zidentyfikowane przez aplikację. Użytkownik posiada także możliwość zmiany danego rozdziału na następny lub poprzedni odpowiednio przyciskiem *Next* lub *Prev*. Wtedy zostanie on poinformowany odpowiednim komunikatem. Statystyka globalna i lokalna aplikacji Machine Book Extract została pokazana na Rys. 6.4.

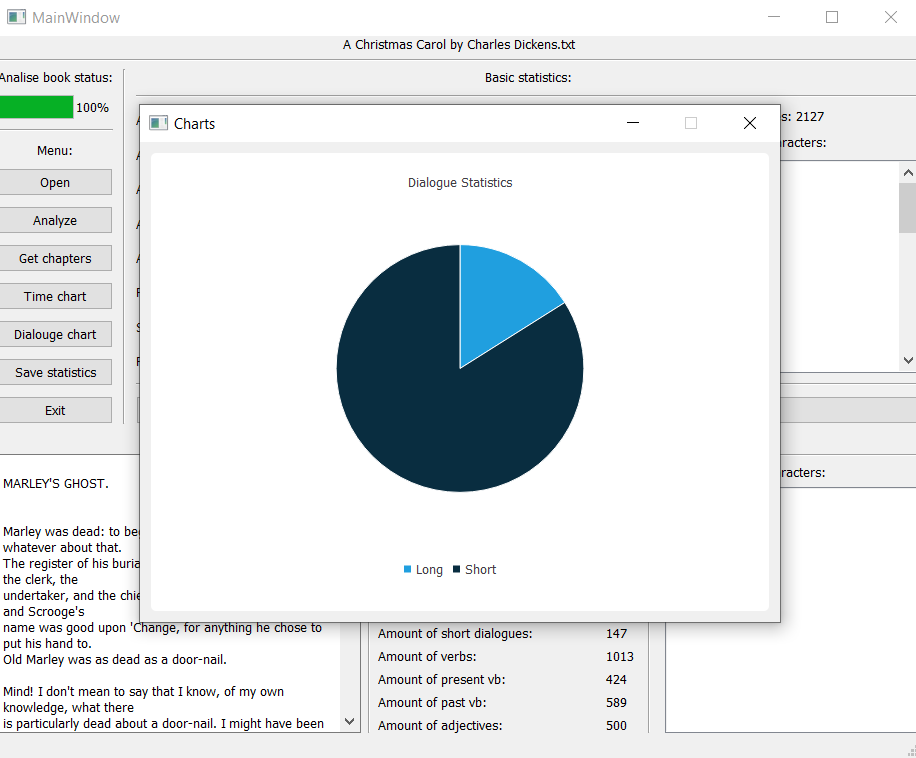


Rys. 6.4 Przykładowe dane uzyskane po analizie pliku tekstowego zawierającego treść książki „Opowieść Wigilijna” Charlesa Dickensa w programie Machine Extract Book

Następnie użytkownik może przeczytać informacje wyciągnięte z utworu i poddać je własnej analizie oraz poprawności, czy się zgadzają z rzeczywistymi danymi zawartymi w treści książki np. poprawni bohaterowie utworu i ich ilość. Następnie do dyspozycji użytkownika dostępne są statystyki graficzne dla czasów zawartych w utworze widoczne na Rys. 6.5 oraz dla ilości dialogowych pokazane na Rys. 6.6.

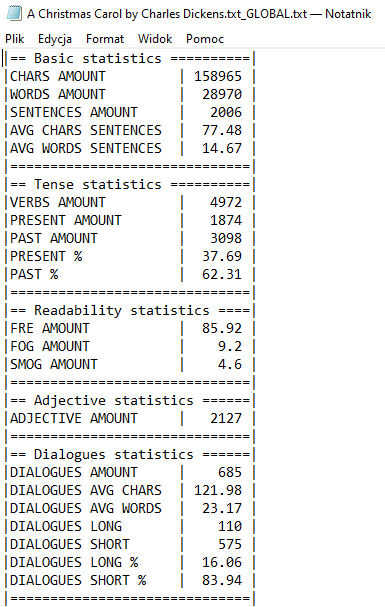


Rys. 6.5 Przykładowy wykres przedstawiający stosunki czasów teraźniejszego do przyszłego w książce „Opowieść Wigilijna” Charlesa Dickensa w programie Machine Extract Book

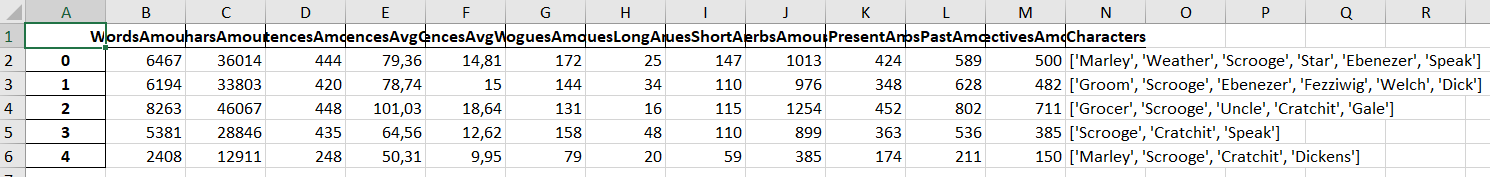


Rys. 6.6 Przykładowy wykres przedstawiający stosunki dialogów długich do krótkich w książce „Opowieść Wigilijna” Charlesa Dickensa w programie Machine Extract Book

Po zamknięciu dowolnego okna ze statystykami, użytkownik ma możliwość jeszcze zapisania statystyk do pliku tekstowego dla wartości globalnych oraz do arkusza kalkulacyjnego dla wartości lokalnych. Po naciśnięciu przycisku *Save Statistics* program zapisze wyciągnięte informacje odpowiednie do obu plików i będą one prezentowały się w formie widocznej na Rys. 6.7 i Rys. 6.8.



Rys. 6.7 Przykładowe statystyki globalne dla książki „Opowieść Wigilijna” Charlesa Dickensa wyciągnięte za pomocą programu Machine Extract Book



Rys. 6.8 Przykładowe statystyki lokalne dla książki „Opowieść Wigilijna” Charlesa Dickensa wyciągnięte za pomocą programu Machine Extract Book

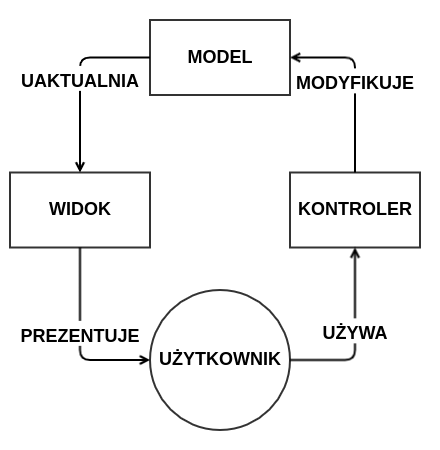
## Zarys implementacji

Nadrzędną klasą odpowiedzialną za funkcje analizy danej książki jest book\_analyser\_global. W niej znajduje się metoda start(), która zaczyna przetwarzanie informacji dla danej książki. W konstruktorze są inicjalizowane moduły związane z biblioteką spaCy dla sylab, NLP języka angielskiego oraz stałe dla maksymalnej ilości znaków podczas przetwarzania danego tekstu, gdyż domyślnie biblioteka spaCy jest skierowana do małej ilości tekstu. Główny moduł zawiera wszystkie narzędzia, które analizują daną książkę pod kątem części mowy w niej występujących, bohaterów, długości książki i jej czytelności. W głównym module są też szukane rozdziały, o ile dana książka je posiada, aby zmniejszyć czas oczekiwania użytkownika do minimum, a nie dzielić go na wiele etapów, odpowiedzialnych za każdy przycisk osobno.

Klasa book\_analyser\_local jest klasą podobną do nadrzędnej, aczkolwiek posiada ograniczoną funkcjonalność ze względu na informacje lokalne wyciągane z danych fragmentów książki. Jest ona użyta w module chapters\_tool, gdzie dla każdego znalezionego fragmentu są wyciągane jego statystyki.

Klasa nadrzędna i podrzędna aplikacji łączą się z modułem zapisu informacji oraz wyświetlania ich w celu szybkiego poinformowania użytkownika o potencjalnych wynikach lub zapisania wyników w celu dalszej analizy danego dzieła.

Całość jest spięta w klasie main\_app, gdzie jest wyświetlane okno danego dzieła oraz ładowane GUI dla danej aplikacji oraz podłączone sterowanie dla poszczególnych funkcjonalności w programie. Całość przypomina wzorzec MVC stosowany w wielu aplikacjach desktopowych. Służy on do organizowania struktury aplikacji tak aby model był reprezentowaną logiką aplikacji (w tym przypadku book\_tools i book\_tools\_main) widok, który opisuje pewną część modelu i interfejsu użytkownika (tutaj znajduje się moduł GUI) przez kontroler, który przyjmuje dane wejściowe od użytkownika i reaguje na poszczególne jego poczynania analizując działania i odświeżając widoki interfejsu (tutaj main\_app) [[32]](#_[Online]_https://pl.wikipedia.org/w_6). Przykładowy wygląd wzorca został przedstawiony na Rys. 6.9.



Rys. 6.9 Wzorzec MVC dla aplikacji okienkowych zawarty w programie Machine Extract Book (https://pl.wikipedia.org/wiki/Model-View-Controller#/media/Plik:Mvc-diagram.png)

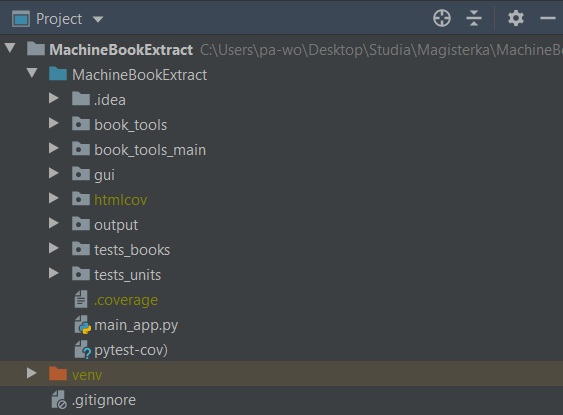
## Opis aplikacji

Główną klasą programu jest book\_analyser\_global. Ze względu na dużą liczbę znaków przetwarzaną przez program, w poniższej aplikacji została ustawiona jego górna granica jako 2,5 mln znaków, aczkolwiek użytkownik rozszerzając funkcjonalność może ją zwiększyć. Parser v2.x wykorzystany w programie wymaga około 1 GB pamięci tymczasowej na 100 000 znaków w danych wejściowych. Oznacza to, że długie teksty mogą powodować błędy alokacji pamięci, dlatego dostępnym maksymalnym domyślnym limitem jest 1 000 000 znaków [[38]](#_[Online]_https://github.com/explosi). Niestety spaCy jest przez to ograniczone i wraz ze wzrostem liczby znaków, dany plik może być dłużej przetwarzany. Klasa book\_analyser\_local jest bardzo podobna do wyżej opisanej, tylko posiada część funkcjonalności dla statystyk lokalnych, które następnie są zapisywane w dataframe z biblioteki pandas. Ostatnią klasą modułu book\_tools\_main jest book\_analyser\_output, który zarządza zapisywaniem plików tekstowych i arkuszy kalkulacyjnych, lub w przypadku debugowania programu wypisywaniem informacji do konsoli.

Moduł book\_tools zawiera narzędzia przydatne w trakcie analizy danego tekstu. Wszystkie opierają się na bibliotece spaCy i wykorzystują dostarczone przez niego funkcje. Klasa adjective\_tool zawiera narzędzia do wyciągania informacji o przymiotnikach w danym tekście, natomiast moduł basic\_statistics\_tool.py zawiera narzędzia do wyciągania podstawowych informacji statystycznych (m.in. liczba znaków i słów). Klasa chapters\_tool zawiera narzędzia do wyciągania informacji z rozdziałów, jeśli takie zostaną znalezione w tekście. Klasa characters\_tool wraz z characters\_person\_rate pozwala za pomocą algorytmu wag wyciągnąć najbardziej pasujących bohaterów znajdujących się w danym tekście w stosunku ilości występowania jak i odpowiedniego znacznika oznaczonego przez bibliotekę spaCy. Dialogue\_tool zawiera narzędzia do obsługi dialogów i informacji w nich zawartych. Klasa read\_statistics\_tool zawiera informacje o współczynnikach czytelności danego tekstu i oblicza je za pomocą odpowiednich wzorów. Na końcu klasa time\_statistics\_tool przetwarza czasowniki zawarte w tekście oraz informacje z nich wyciągnięte.

Moduł GUI zawiera wygląd okienkowy danej klasy oraz wielowątkowość, tak aby program mógł przeprowadzać analizę danego dzieła i nie blokować działań użytkownika w przypadku, jeśli analizuje on poprzednie dzieło).

Folder tests\_books zawiera książki poddane testom i informacje z nich wyciągnięte. Podobnie tests\_units posiada testy, natomiast są w nich zawarte testy jednostkowe oraz pokrycie testami, które zostało omówione w rozdziale 6.6. Widok struktury projektu został przedstawiony na Rys. 6.10.



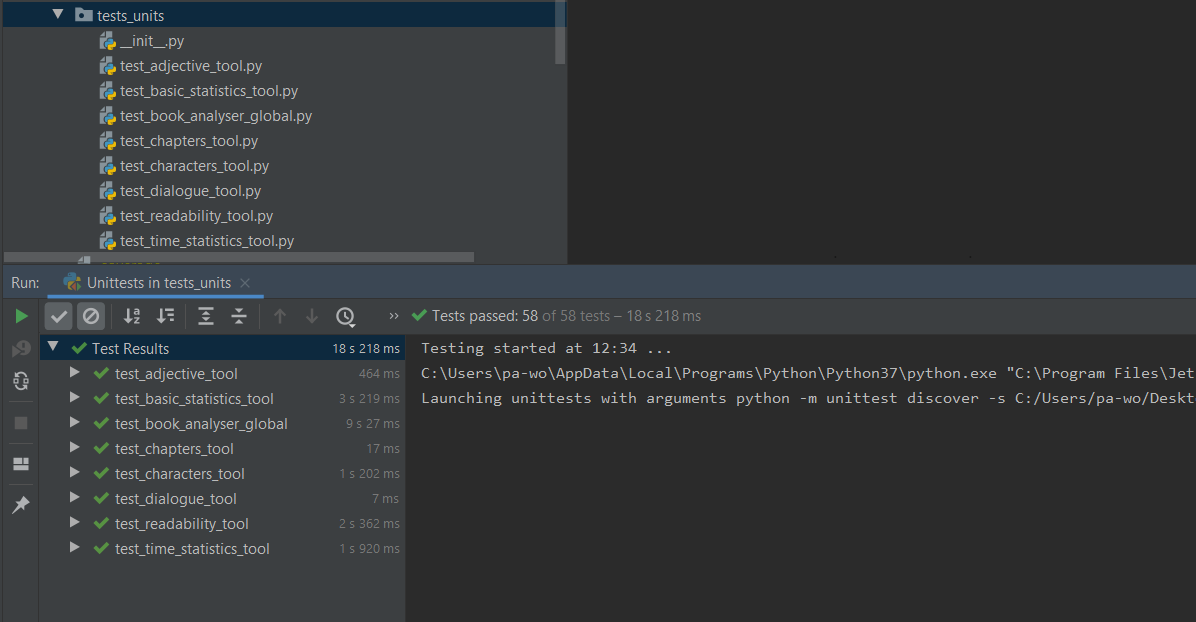
Rys. 6.10 Struktura programu Machine Extract Book

## Testy aplikacji

Narzędzia unittests służą do tworzenia testów jednostkowych napisanego oprogramowania zainspirowane przez JUnit i mający podobny wygląd. Obsługuje także automatyzację testów udostępniając kod konfiguracji, wyłączenie testów i ich agregację oraz niezależność testów od struktury raportowania. Różne przypadki testowe są w stanie sprawdzić każdy aspekt i zachowanie kodu. Dzięki nim można oddzielić testowanie od samego kodu [[23]](#_[Online]_https://docs.python.org/3/).

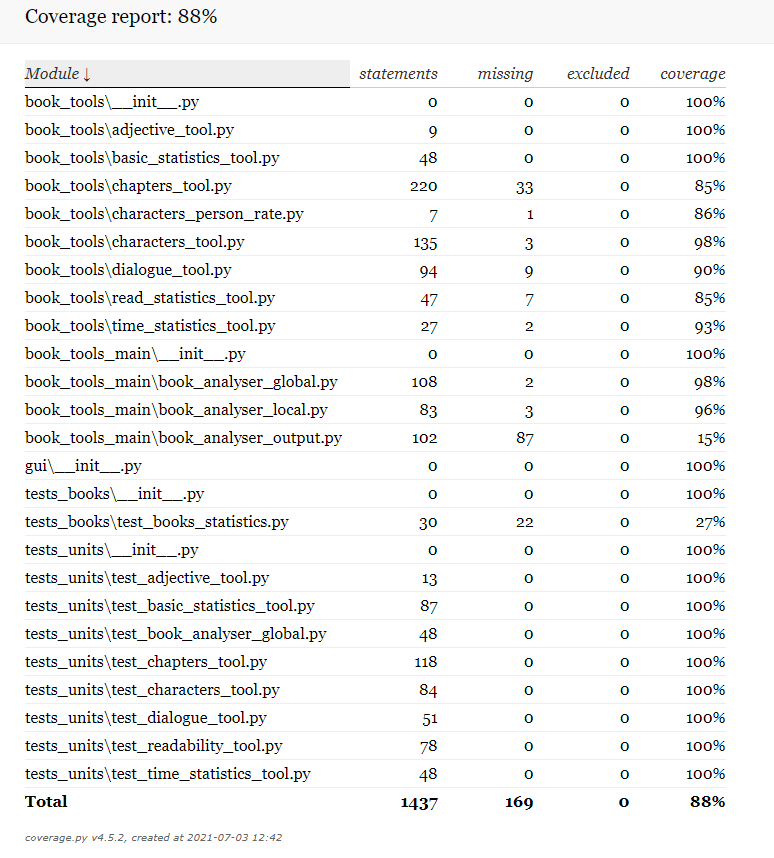
Każdy test powinien posiadać tzw. osprzęt testowy, czyli przygotowanie wszystkich niezbędnych składników do przeprowadzenia danego testu. Kolejnym niezbędnym składnikiem jest przypadek testowy, czyli jednostkę testującą sprawdzającą reakcję na określony zestaw danych podanych wejściowo przez użytkownika lub program. Unittest udostępnia klasę TestCase, która służy do tworzenia przypadków testowych. Ponadto biblioteka zawiera zestaw przypadków testowych, który służy do agregowania testów, które powinny być wykonywane razem. Na końcu występuje program uruchamiający testy jako składnik, który koordynuje wykonywanie testów i udostępnia wynik użytkownikowi, aby wskazać wynik wykonywanych testów.

Dla aplikacji Machine Extract Book zostało przygotowane 58 testów sprawdzających narzędzia i moduły do wyciągania informacji z utworów literackich. Przykładowy wygląd i wykonanie testów zostały przedstawione na Rys. 6.11.



Rys. 6.11 Widok po uruchomieniu testów w aplikacji Machine Extract Book

Ponadto dla przetestowania pokrycia kodem został użyta biblioteka coverage.py. Jest to narzędzie do pokrycia kodu programów języka Python. Monitoruje on program odnotowując, które części zostały wykonane, a następnie analizuje źródło, aby zidentyfikować kod, który został wykonany, a który nie. W przypadku podania źródła jako testów, aplikacja pokazała jakie części kodu zostały przetestowane. Pokrycie kodem zostało użyte z poziomu pliku *src* i linii komend, a widoczne pokrycie kodem zostało przedstawione na Rys. 6.12.



Rys. 6.12 Widok pokrycia testem klas w aplikacji Machine Extract Book

Pokrycie kodem wskazuje 88 procent. Niektóre moduły nie mogły zostać przetestowane ze względu na użycie zewnętrznych bibliotek oraz wyglądu użytkownika. Dane pokrycie mogłoby zostać zwiększone poprzez ograniczenie niektórych funkcjonalności, ale nie dałoby to wtedy informacji przydatnej przy analizie i testowaniu wielu książek.

Prócz testów jednostkowych, zostały także wykonane testy poprawności wyciągania informacji z danej książki przy sprawdzeniu jakości i poprawności działania algorytmów. Przy większej ilości bohaterów w tekście lub bohaterów fikcyjnych np. Królik w książce „Alicja w Krainie Czarów”, algorytm nie jest w stanie sobie poradzić czy jest to nazwa bohatera czy też zwierzęcia, które występuje w książce. Podobnie przy braku bohaterów w książce, algorytm może sugerować „na siłę” ich propozycje, które mogą nie być poprawne.

Podobnie przy próbie wykrycia czy dany utwór literacki dzieli się na rozdziały, w przypadku podziału książki w sposób nie spełniający kryteria programu, opisane w rozdziale 5, dzieło nie zostanie prawidłowo podzielone. Przykładowo przy braku odstępów po linijkach zawierających nazwy rozdziałów lub ich dowolnej numeracji, algorytm może nie zadziałać poprawnie stwierdzając, iż w danym utworze mogą nie występować rozdziały.

Podczas testowania zdarzało się też, iż niepoprawnie zostały wyciągnięte dialogi. Było to spowodowane również inwencją twórczą autora, gdyż nie podał ich w cudzysłowie ani bez myślnika lub nowej linijki. Sam czytelnik mógłby zostać wprowadzony w błąd czy faktycznie napotkał na dialog. Program nie jest w stanie wyciągnąć poprawnie dialogów bez spełnienia kryteriów opisanych wyżej.

SpaCy także przy długich tekstach i napisanych skomplikowanym słownictwem nie zawsze jest w stanie dobrze oznaczyć daną część mowy. Podobna sytuacja ma miejsce w przypadku bardzo długich zdań, gdzie algorytm może pogubić się przy dużej zależności między słowami. Dzieje się też tak gdy występują nazwy własne zmodyfikowane przez autora. SpaCy może wtedy błędnie oznaczyć daną część mowy, co za tym idzie w algorytmach stworzonych do przetwarzania informacji z danej książki mogą wystąpić błędy.

Przedstawione algorytmy są wrażliwe na treść zawartą w danym utworze. Jeśli utwór jest innowacyjny, ma nietypowy gatunek literacki bądź nie wpasowuje się w schemat zawarty w większości utworów, wtedy algorytmy mogą nie przynieść poprawnych rezultatów. Może się też zdarzyć, iż będą działały poprawnie, natomiast analiza informacji wyciągniętych za pomocą programu będzie dłuższa dla danego użytkownika. Algorytmy zostały wykonane bez uczenia maszynowego (machine learning), więc parametry do przetwarzania danych zawartych w dziełach literackich bazują na zbiorze testowych książek:

* „Opowieść Wigilijna” autorstwa Charlesa Dickensa
* „Alicja w Krainie Czarów” napisana przez Charlesa Lutwidge’a Dodgsona
* „Drakula” Brama Stokera
* „Metamorphosis” autorstwa Franza Kafki
* „Moby Dick albo Wieloryb” napisana przez Hermana Melville’a
* „Piotruś Pan” autorstwa Jamesa Matthew Barrie
* „Duma i uprzedzenie” powieść Jane Austen
* „Romeo i Julia” autorstwa Williama Szekspira
* „Przygody Sherlocka Holmesa” napisana przez Arthura Conana Doyle’a
* „Zamczysko w Otranto” autorstwa Horacego Walpole'a
* „Wichrowe Wzgórza” Emily Jane Brontë
* „Sonety” autorstwa Williama Szekspira
* „The History of Freedom and Other Essays” autorstwa Johna Emericha Edwarda Dalberga-Actona

# Wnioski

## Wnioski badawcze

W powyższej pracy zostały pokazana metoda ekstrakcji informacji z utworu literackiego. Jest to bardzo obszerna dziedzina analizy i oprogramowania, która nadal się rozwija, aby przynieść jak najlepsze efekty i zrozumienie tekstu przez maszynę. Języki przetwarzania języka maszynowego dostarczają wiele funkcjonalności i są w stanie przeanalizować wstępnie dany tekst i ograniczyć czas spędzony nad danym utworem przez użytkownika. Są także pomocne przy dalszych etapach analizy tekstu i niosą ze sobą wiele ważnych informacji. Powstała aplikacja zawierająca wstępny proces analizy dzieła literackiego, umożliwiająca wyciąganie statystyk globalnych i lokalnych oraz przedstawianie ich reprezentacji graficznej, a także dająca możliwość zapisania i edycji otrzymanych informacji.

Użytkownicy korzystając z danej aplikacji mogą prowadzić badania nad utworami literackimi i wymieniać się wiedzą, a także poddawać masowej obróbce zawartości dzieł, aby wyciągnąć z nich wstępne informacje. Mogą być zamieszczane i publikowane, aby proces czytania i zrozumienia danego dzieła został przyśpieszony i maksymalnie wykorzystany dla danego utworu.

## Drogi rozwoju aplikacji

Tematyka porusza obszar nauczania języka maszynowego, który jest nadal rozwijany. Wraz ze zwiększeniem funkcjonalności opisanej w pracy bibliotek, aplikacji może nieść ze sobą coraz doskonalsze wyniki, ale również być dużo bardziej optymalna dla dłuższych dzieł literackich. Kod jest zamknięty na modyfikacje, ale i otwarty na rozszerzenia, dzięki czemu rozwój danego oprogramowania w przyszłości będzie możliwy i prosty do wprowadzenia. Jedną z dróg rozwoju jest użycie algorytmów opracowanej aplikacji w popularnych serwisach zawierających długie artykuły internetowe, aby została opracowana ich analiza, ale także i streszczenie, które może być końcowym etapem poruszanego w pracy problemu.

# Literatura

### E. D. Liddy, „Natural language processing”, Encyklopedia Bibliotekoznawstwa i Informacji Naukowej, Wydawnictwo Nowy Jork. Marcel Decker, 2001

### [Online] <https://ai.pwn.pl/blog/nlp-co-to-jest-i-do-czego-sie-przydaje>

### A. M. Turing, „Computing machinery and intelligence”, Dordrecht, 2009

### Y. Bar-Hillel, „The present status of automatic translation of languages”, Wydawnictwo MIT Press, 2003

### [Online] <https://monkeylearn.com/sentiment-analysis/>.

### [Online] <https://pl.wikipedia.org/wiki/SHRDLU>

### [Online] <https://en.wikipedia.org/wiki/ALPAC>

### [Online] <https://en.wikipedia.org/wiki/Jabberwacky>

### [Online] <https://pl.wikipedia.org/wiki/Siri>

### [Online] <https://en.wikipedia.org/wiki/Tokenization_(data_security)>

### [Online] <https://en.wikipedia.org/wiki/Lemmatisation>

### [Online] https://pl.wikipedia.org/wiki/Stemming

### [Online] <https://tokeneo.com/pl/czym-jest-tokenizacja-wszystko-co-powinienes-wiedziec-na-jej-temat/>

### M. P. Marcus, B. Santorini and M. A. Marcinkiewicz, „Building a large annotated corpus of English: The Penn Treebank”, Computational Lingustics, 19(2), pp. 313-330, 1993.

### [Online] <https://en.wikipedia.org/wiki/Parsing>

### [Online] <https://ichi.pro/pl/analiza-nastrojow-modele-leksykonow-a-uczenie-maszynowe-201089019275078>

### [Online] <https://aws.amazon.com/textract/>

### [Online] <https://hbsp.harvard.edu/product/620025-PDF-ENG>

### [Online] <https://ir.taskus.com/>

### [Online] <https://em360tech.com/top-10/top-10-natural-language-processing-solutions>

### [Online] <https://medinreal.com/>

### [Online] <https://pl.wikipedia.org/wiki/Python>

### [Online] <https://docs.python.org/3/library/unittest.html>

### [Online] <https://spacy.io/usage/spacy-101>

### [Online] <https://pypi.org/project/PyQt5/>

### [Online] <https://www.qt.io/product>

### [Online] <https://www.masswerk.at/elizabot/>

### [Online] <https://klp.pl/syzyfowe-prace/ser-5.html>

### [Online] <https://pl.wikipedia.org/wiki/Indeks_czytelno%C5%9Bci_Flescha>

### [Online] <https://pl.wikipedia.org/wiki/Indeks_czytelno%C5%9Bci_FOG>

### [Online] <https://en.wikipedia.org/wiki/SMOG>

### [Online] <https://pl.wikipedia.org/wiki/Model-View-Controller>

### [Online] https://en.wikipedia.org/wiki/ELIZA

### [Online] https://en.wikipedia.org/wiki/GPT-3

### [Online] https://copilot.github.com/

### [Online] <https://www.gutenberg.org/files/46/46-h/46-h.htm>

### [Online] https://www.gutenberg.org/ebooks/31278

#### [Online] https://github.com/explosion/spaCy/blob/master/spacy/errors.py

#### [Online] https://towardsdatascience.com/a-practitioners-guide-to-natural-language-processing-part-i-processing-understanding-text-9f4abfd13e72

# Spis ilustracji

[Rys. 3.1 Joseph Weizenbaum przy programie ELIZA. (https://www.masswerk.at/elizabot/) 13](#_Toc79746978)

[Rys. 4.1 Hierarchia typów obecna w języku Python. (https://commons.wikimedia.org/wiki/File:Python\_3.\_The\_standard\_type\_hierarchy.png) 21](#_Toc79746979)

[Rys. 4.2 Przykład rozbioru zdania „Apple is looking at buying U.K. startup for $1 billion” przez bibliotekę spaCy (https://spacy.io/usage/linguistic-features) 23](#_Toc79746980)

[Rys. 4.3 Przykładowe znaczniki części mowy używane przez bibliotekę spaCy (https://newbedev.com/what-do-spacy-s-part-of-speech-and-dependency-tags-mean) 24](#_Toc79746981)

Rys. 4.4 Wygląd aplikacji Qt Designer dla tworzenia aplikacji okienkowej (https://pl.wikipedia.org/wiki/PyQt#/media/Plik:Screenshot-qt-designer-qwebview.png) 26

[Rys. 5.1 Przykładowe implementacja algorytmu wyboru bohaterów utworu w aplikacji Machine Extract Book 29](#_Toc79746983)

[Rys. 6.1 Główne okno programu Machine Extract Book 50](#_Toc79746984)

[Rys. 6.2 Okno otwierania pliku do przetwarzania w programie Machine Extract Book 51](#_Toc79746985)

[Rys. 6.3 Proces przykładowej analizy pliku tekstowego zawierającego treść książki „Opowieść Wigilijna” Charlesa Dickensa w programie Machine Extract Book 52](#_Toc79746986)

[Rys. 6.4 Przykładowe dane uzyskane po analizie pliku tekstowego zawierającego treść książki „Opowieść Wigilijna” Charlesa Dickensa w programie Machine Extract Book 53](#_Toc79746987)

[Rys. 6.5 Przykładowy wykres przedstawiający stosunki czasów teraźniejszego do przyszłego w książce „Opowieść Wigilijna” Charlesa Dickensa w programie Machine Extract Book 54](#_Toc79746988)

[Rys. 6.6 Przykładowy wykres przedstawiający stosunki dialogów długich do krótkich w książce „Opowieść Wigilijna” Charlesa Dickensa w programie Machine Extract Book 54](#_Toc79746989)

[Rys. 6.7 Przykładowe statystyki globalne dla książki „Opowieść Wigilijna” Charlesa Dickensa wyciągnięte za pomocą programu Machine Extract Book 55](#_Toc79746990)

[Rys. 6.8 Przykładowe statystyki lokalne dla książki „Opowieść Wigilijna” Charlesa Dickensa wyciągnięte za pomocą programu Machine Extract Book 55](#_Toc79746991)

[Rys. 6.9 Wzorzec MVC dla aplikacji okienkowych zawarty w programie Machine Extract Book (https://pl.wikipedia.org/wiki/Model-View-Controller#/media/Plik:Mvc-diagram.png) 57](#_Toc79746992)

[Rys. 6.10 Struktura programu Machine Extract Book 58](#_Toc79746993)

[Rys. 6.11 Widok po uruchomieniu testów w aplikacji Machine Extract Book 59](#_Toc79746994)

[Rys. 6.12 Widok pokrycia testem klas w aplikacji Machine Extract Book 60](#_Toc79746995)

Załącznik nr 1

do Procedury antyplagiatowej prac dyplomowych

w Politechnice Łódzkiej

Łódź, dn. ......……………….

**OŚWIADCZENIE**

Paweł Wolski

………………………………………………………….

*(Imię i nazwisko studenta)*

Orla 3 m. 10, 90-317 Łódź

………………………………………………………….

*(Adres)*

234014

………………………………………………………….

*(Nr albumu)*

Wydział Elektrotechniki, Elektroniki,

Informatyki i Automatyki

………………………………………………………….

*(Jednostka organizacyjna prowadząca studia)*

Informatyka

………………………………………………………….

*(Kierunek studiów)*

Studia II stopnia magisterskie stacjonarne

………………………………………………………….

*(Poziom kształcenia i forma studiów)*

Oświadczam, że poinformowano mnie o zasadach dotyczących kontroli oryginalności pracy dyplomowej w Jednolitym Systemie Antyplagiatowym.

.………………………….......

*(Podpis studenta)*

Załącznik nr 2

do Procedury antyplagiatowej prac dyplomowych

w Politechnice Łódzkiej

Łódź, dn. ......……………….

**OŚWIADCZENIE**

**o samodzielności wykonania i oryginalności pracy dyplomowej**

Paweł Wolski

………………………………………………………….

*(Imię i nazwisko studenta)*

Orla 3 m. 10, 90-317 Łódź

………………………………………………………….

*(Adres)*

234014

………………………………………………………….

*(Nr albumu)*

Wydział Elektrotechniki, Elektroniki,

Informatyki i Automatyki

………………………………………………………….

*(Jednostka organizacyjna prowadząca studia)*

Informatyka

………………………………………………………….

*(Kierunek studiów)*

Studia II stopnia magisterskie stacjonarne

………………………………………………………….

*(Poziom kształcenia i forma studiów)*

Świadomy/a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że:  
1. Przedkładana praca dyplomowa **magisterska** [[1]](#footnote-1)\*) na temat: **Ekstrakcja informacji z utworu literackiego** została wykonana przeze mnie samodzielnie.

2. Wymieniona wyżej praca:

- nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz. U. z 2018 poz. 1191, z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym, a także nie zawiera danych i informacji, które uzyskałem/am w sposób niedozwolony,

- nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów wyższej uczelni lub tytułów zawodowych.

3. Jednocześnie wyrażam zgodę/nie wyrażam zgody\*\* na wykorzystanie fragmentów mojej pracy dyplomowej w publikacjach naukowych pracowników Politechniki Łódzkiej za zgodą prodziekana właściwego ds. studenckich, na zasadach wynikających z Ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz. U. z 2018 poz. 1191, z późn. zm.)

.………………………….......

*(Podpis studenta)*

Załącznik nr 3

do Procedury antyplagiatowej prac dyplomowych

w Politechnice Łódzkiej

Łódź, dn. ......……………….

**OŚWIADCZENIE**

**o zgodności wersji elektronicznej pracy dyplomowej  
z przedstawionym wydrukiem komputerowym**

Paweł Wolski

………………………………………………………….

*(Imię i nazwisko studenta)*

Orla 3 m. 10, 90-317 Łódź

………………………………………………………….

*(Adres)*

234014

………………………………………………………….

*(Nr albumu)*

Wydział Elektrotechniki, Elektroniki,

Informatyki i Automatyki

………………………………………………………….

*(Jednostka organizacyjna prowadząca studia)*

Informatyka

………………………………………………………….

*(Kierunek studiów)*

Studia II stopnia magisterskie stacjonarne

………………………………………………………….

*(Poziom kształcenia i forma studiów)*

Świadomy odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że przedkładana na nośniku elektronicznym praca dyplomowa **magisterska** [[2]](#footnote-2)\*) na temat: **Ekstrakcja informacji z utworu literackiego** zawiera te same treści, co oceniany przez promotora i recenzenta wydruk komputerowy.

Jednocześnie oświadczam, że jest mi znany przepis art. 233 § 1 Kodeksu karnego określający odpowiedzialność za składanie fałszywych zeznań.

.………………………….......

*(Podpis studenta)*

1. \*) Wpisać odpowiednio: licencjacka, inżynierska, magisterska.  
   \*\*) niepotrzebne skreślić [↑](#footnote-ref-1)
2. \*) Wpisać odpowiednio: licencjacka, inżynierska, magisterska. [↑](#footnote-ref-2)