

SPIS TREŚCI

Nowa wiedza wytworzona w projekcie.....	3
Definicja problemu	4
Analiza dotychczasowego dorobku naukowego	5
Przyjęta metodologia projektowa	7
Uzyskane rezultaty.....	10
Spis literatury	12
Działanie 1 (badania przemysłowe) - Opracowanie metod przetwarzania tekstu pod kątem ekstrakcji wyrażen reprezentujących reguły biznesowe	13
1. Przygotowanie narzędzi i infrastruktury do tworzenia i zarządzania zbiorem uczącym. 13	
2. Opracowanie i adnotowanie zbioru uczącego.	15
3. Ewaluacja i wybór podstawowych bibliotek przetwarzania tekstu (standaryzacja morfologiczna, lematyzacja, embedding).	38
4. Ewaluacja narzędzi, badania i wybór metody klasyfikacji semantycznej zdań.....	46
5. Przeprowadzenie właściwych badań mających na celu ustalenie właściwego potoku przetwarzania danych, wybór struktury modelu i parametryzacja algorytmów	50
Opis wyniku działania ewaluatora reguł i pozyskana nowa wiedza:	82
Działanie 2 (badania przemysłowe) - Opracowanie metod translacji reguł biznesowych wyrażonych w tekście na reguły wykonywalne, opracowanie silnika ewaluacji reguł.	84
1. Opracowanie algorytmu wiązania dokumentów w kontekst biznesowy na podstawie treści dokumentów	84
2. Opracowanie języka modelowania reguł biznesowych opartego na składni języka naturalnego.....	94
3. Opracowanie algorytmu mapowania wyekstrahowanych z dokumentów wyrażen deontycznych na szablony reguł biznesowych.	100
4. Implementacja ewaluatora reguł w kontekście zbioru powiązanych dokumentów. ..	111
Działanie 3 (prace rozwojowe).....	115

Nowa wiedza wytworzona w projekcie

Beneficjent oświadcza, że w trakcie realizacji projektu przeprowadzone zostały wszystkie przewidziane w projekcie prace zarówno w fazie badań przemysłowych jak i prac rozwojowych. Jednocześnie trzeba zauważyć, że zobowiązaniem beneficjenta było - zgodnie z narzuconą umową metodyką projektową - osiągnięcie założonych w projekcie kamieni milowych. Umowa o dofinansowanie nie definiuje szczegółowych wymogów dotyczących sposobu dokumentowania prowadzonych prac badawczo-rozwojowych. Beneficjent nie jest też jednostką naukową ukierunkowaną na publikację artykułów naukowych, a komercyjnym podmiotem ukierunkowanym na końcowy wynik, jakim jest osiągnięcie celów projektu (kamieni milowych), w tym ostatecznie prototypu produktu, który ma potencjał do wdrożenia produkcyjnego oraz komercjalizacji na rynku.

Ze względu na specyfikę działalności beneficjenta (komercyjne wytwarzanie oprogramowania) w projekcie stosowane było iteracyjne podejście do rozwoju oprogramowania, które zakłada ciągle usprawnianie uzyskanych rozwiązań w kolejnych iteracjach, przy czym pośrednie wyniki są tylko efektem chwilowym stanu rozwiązania na daną iterację i jako takie nie są dokumentowane gdyż w iteracyjnym modelu wytwarzania istotne jest osiągnięcie celów projektu (kamieni milowych), a nie dokumentowanie działań, które w kolejnych iteracjach są "nadpisywane" przez rozwiązania, które przybliżać mają projekt do osiągnięcia założonego celu. Jest to zgodne z metodyką Agile, która jest powszechnie uznana za prawidłową przez całą branżę IT, w szczególności jest to zgodne z zasadą "Working software is the primary measure of progress" (źródło: <https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/>)

Takie podejście jest zgodne z założeniami umowy o dofinansowanie, która jest zorientowana na definicję i weryfikację kamieni milowych poszczególnych etapów oraz prawidłowe wydatkowanie środków, co potwierdzają odpowiednie dokumenty finansowe. Zarówno kamienie milowe, opis działań jak i skład zespołu projektowego został zaakceptowany przez instytucję finansującą w momencie przyznania dotacji oraz na etapie realizacji projektu i beneficjent postępował przez cały projekt zgodnie z tymi opisami wykorzystując jednocześnie metodykę Agile, jako przyjętą w branży najlepszą praktykę branżową. Takie ukierunkowanie na cele projektu może być źródłem dysonansu poznawczego dla ekspertów oceniających raport, którzy oczekują bardziej szczegółowej dokumentacji poszczególnych działań i wyników, jednak należy zwrócić uwagę, że beneficjentowi nie przedstawiono dokładnych wytycznych dotyczących sposobu dokumentowania prac. Realizacja tych wymagań w momencie, kiedy od czasu realizacji niektórych zadań minęły 2 lata jest trudna i wymaga konieczności zaangażowania zasobów w przywracanie stanu projektu z wcześniejszych iteracji, co nie zawsze jest w 100% możliwe. Chcąc jednak wyjść naprzeciw oczekiwaniom ekspertów przedstawiamy poniżej bardziej precyzyjny opis wykonanych kroków i sposobu wnioskowania oraz doprowadzającego do ostatecznych rozwiązań, które znalazły ostatecznie zastosowanie w projekcie, na tyle na ile jest to możliwe do wykonania post factum.

Definicja problemu

Zgodnie z treścią wniosku o dofinansowanie celem projektu było opracowanie algorytmów umożliwiających ekstrakcję reguł biznesowych z treści tekstów dokumentów oraz opracowanie ewaluatora tych reguł, który będzie umożliwiał weryfikację, czy inne powiązane dokumenty są poprawne w kontekście zidentyfikowanych reguł. Problem ten jest związany z interpretacją tzw. wyrażen deontycznych. Wyrażenia te mają strukturę wyrażającą w języku naturalnym pewne reguły, które mogą być spełnione lub nie w momencie wystąpienia pewnych zdarzeń. Ewaluacja reguł sprowadza się do wyliczenia wartości logicznej prawda/fałsz w zależności od wartości zmiennych związanych z danym zdarzeniem, które są ewaluowane zgodnie z definicją algorytmu reguły. W pracach naukowych na ten temat zazwyczaj przyjmowana jest pewna klasyfikacja tych reguł w zależności od tego, jaki wpływ ma ich realizacja na zdarzenia występujące w danym kontekście. Modalność reguł w najbardziej ogólnym ujęciu dzieli reguły na reguły permissive (dopuszczające pewne zdarzenia), obligatoryjne (określające, że pewne zdarzenia muszą wystąpić) i niezmienniki, które muszą zawsze być prawdziwe. Z punktu widzenia języka naturalnego najbardziej ogólna struktura zdań wyrażająca reguły ma formę podmiot - relacja - obiekt, np "wykonawca zapłaci kwotę 100PLN", "podatek VAT wynosi 23%" etc.

Ze względu na charakterystyczny dla Beneficjenta obszar działania zgodnie z zapisami wniosku przyjęto, że kontekst dziedzinowy, której będzie rozważany będzie dotyczył przede wszystkim walidacji reguł określonych w umowach definiujących zobowiązania wobec kontrahentów w odniesieniu do faktur wystawianych pomiędzy kontrahentami. Dodatkowym założeniem było to, że rozwiązanie ma przede wszystkim działać dla języka polskiego, co jest istotne dla rynku, na którym działa Beneficjent. Tak zdefiniowany problem był rozwiązywany za pomocą technik przetwarzania języka naturalnego (NLP). Potencjalne techniki możliwe do użycia zostały określone we wniosku o dofinansowanie, ale dopiero przeprowadzenie prac badawczych w ramach projektu wskazało zarówno ślepe uliczki jak i rozwiązania, które okazały się skuteczne. Główna trudność w określeniu wstępnych założeń do projektu związana była przede wszystkim z następującymi problemami:

1. Trudność z dostępem do literatury naukowej dotyczącej prac tego typu w języku polskim. Większość prac dotyczących zagadnienia ekstrakcji reguł deontycznych dotyczy języka angielskiego i chińskiego.
2. Trudności z dostępem do literatury dotyczącej ewaluacji reguł w dziedzinie relacji biznesowych, gdyż zdecydowana większość prac z obszaru ekstrakcji reguł deontycznych dotyczy normatywnych aktów prawnych i ich analizy pod kątem budowania baz wiedzy, walidacji spójności reguł etc., a nie problemu ewaluacji w kontekście wystąpienia konkretnych zdarzeń biznesowych.
3. Brak ogólnodostępnych zbiorów danych, które mogą być użyte do przeprowadzenia eksperymentów.

Jak widać już na samym starcie podejmowane zagadnienia wymagały zdobycia nowej, wcześniej niedostępnej wiedzy.

Analiza dotychczasowego dorobku naukowego

W pracy “On Rule Extractions From Legal Regulations” autorzy rozważają możliwości ekstrakcji reguł z dokumentów normatywnych dostępnych w “natywnej” postaci cyfrowej w formacie XML. Materiałem wejściowym są dokumenty w języku angielskim. Celem ekstrakcji jest ewaluacja reguł, a raczej podsumowywanie dokumentów pod kątem eksponowania najważniejszych elementów treści. Wynikiem prac autorów jest opracowanie klasyfikatora, który wybiera z tekstu najważniejsze reguły i przypisuje je do jednej z kilku kategorii. Używane metody sprowadzają się do parsowania struktury zdań za pomocą biblioteki Stanford NLP, próby uogólnienia struktury gramatycznej m.in. zdań poprzez ujednolicanie synonimów i klasyfikacje tak uzyskanych wektorów. Klasyfikacja odbywa się przez dopasowanie do wzorców, które zostały określone ręcznie na podstawie heurystyk. Warto wspomnieć, że w tej pracy autorzy wskazują, że kluczowe dla powodzenia zadania jest zidentyfikowanie tzw. “agentów i tematów” reprezentujących koncepty w zdaniach, a w następnej kolejności identyfikacja kluczowych czasowników wyrażających modalność deontyczną. Autorzy wskazują także, że dużym problemem jest używanie bibliotek, które są trenowane na ogólnie dostępnych tekstach, takich jak artykuły internetowe, do analizy tekstów prawniczych o innej, bardziej skomplikowanej strukturze gramatycznej.

Podobne podejście prezentują autorzy pracy “Combining NLP Approaches for Rule Extraction from Legal Documents”. Główna koncepcja przedstawiona przez autorów zakłada wyjście od z góry założonej lekkiej ontologii, która jest przygotowana ręcznie w postaci heurystyk, które opisują struktury gramatyczne typowe dla wyrażen deontycznych. Te struktury są następnie uogólniane przez zastosowanie synonimów WordNet-u. Podstawową wykorzystywaną biblioteką jest tutaj również Stanford NLP.

W pracy “Agent-Specific Deontic Modality Detection in Legal Language” autorzy również podejmują temat ekstrakcji i klasyfikacji fragmentów tekstu jako reguł deontycznych. Autorzy traktują ten problem wprost jako problem klasyfikacji wieloklasowej. Jednak idą dalej w porównaniu do wcześniejszych prac i metody regułowe traktują tylko jako tzw. “benchmark bazowy” z którym porównują różnego rodzaju klasyfikatory wywodzone z modeli bazujących na transformerach (BERT). Zaproponowane przez autorów klasyfikatory rzeczywiście osiągają w pewnych przypadkach lepszą skuteczność niż metoda bazująca na dopasowaniu reguł. Jednak spektrum porównywanych algorytmów jest wąskie i nie wiadomo dlaczego nie zostały użyte inne rodzaje prostszych (mniej złożonych obliczeniowo) klasyfikatorów np. takie, jakie były wykorzystywane w projekcie Beneficjenta.

Szereg dostępnych prac, jak np. “Automated Directive Extraction from Policy Texts”, “Automated Knowledge Extraction from the Federal Acquisition Regulations System (FARS)”, “Thesaurus Enhanced Extraction of Hohfeld’s Relations from Spanish Labour Law”, “Cognitively Rich Framework to Automate Extraction and Representation of Legal Knowledge”. Wszystkie te prace proponują różne rozwiązania problemu ekstrakcji reguł deontycznych zasadniczo sprowadzając ten problem do problemu klasyfikacji. Różnią się one między sobą rodzajem proponowanych klasyfikatorów oraz mniej lub bardziej pracowitym definiowaniem bazowych ontologii i powiązanych z nimi heurystyk.

Praca “Cognitively Rich Framework to Automate Extraction and Representation of Legal Knowledge” różni się od pozostałych prac tym, że podejmuje próbę ewaluacji wyekstrahowanych reguł w kontekście pewnych zdarzeń (spraw, ang. cases). Autorzy tej pracy podobnie jak Beneficjent przyjęli, że ewaluacja reguł musi się sprowadzać do podstawiania odpowiednich zmiennych w miejscu zidentyfikowanych obiektów (faktów) i wyliczania wartości logicznych wyrażeń. Rodzaje wyrażeń do ewaluacji są przyjęte na podstawie klasyfikacji modalności deontycznej (permissywna, obligatoryjna itd.), a sama klasyfikacja również wykorzystuje dopasowanie słów kluczowych w strukturach zdań.

Praca “Automatic Detection and Semantic Formalisation of Business Rules” idzie dalej w kierunku nie tylko ekstrakcji ale także ewaluacji reguł. Zakres dziedzinowy dotyczy przepisów budowlanych. Zasadniczo koncepcja autorów sprowadza się znów do sformułowania wyjściowej “ontologii” w postaci szablonów reguł w języku SPARQL i powiązanych z nimi list słów kluczowych. Dana reguła jest stosowana jeśli zostanie odnaleziony tekst o odpowiedniej strukturze gramatycznej, w której występuje wymagane słowo kluczowe lub jego synonim. Następnie ekstrahowane są obiekty (fakty), które są podstawiane do reguły. Z kolei w pracy “Agent-Oriented Business Rules: Deontic Assignments” autorzy założyli, że ekstrahowane reguły biznesowe mogą być w sposób formalny wyrażone w języku SQL lub w celu ewaluacji mogą być konwertowane do instrukcji IF-THEN-ELSE, które są dostępne praktycznie w każdym języku programowania.

Praca “The Semantic of Business Vocabulary and Business Rules: An Automatic Generation From Textual Statements” jest pracą, która w najbardziej kompleksowy sposób podejmuje temat ekstrakcji i ewaluacji reguł biznesowych czerpiąc jednocześnie z wcześniejszych dokonań. Przyjęta metoda ekstrakcji zakłada dopasowanie struktury gramatycznej parsowanego tekstu do zestawu z góry przyjętych gramatycznych wzorców heurystycznych. Następnie w wybranych fragmentach tekstu identyfikowane są kluczowe obiekty i czasowniki i na tej podstawie generowane są wyrażenia w języku SVBR.

Z powyższej analizy istniejącego dorobku naukowego wynikają następujące wnioski, które potwierdzają zasadność i aspekt nowości przeprowadzonych prac projektowych:

1. Istniejąca literatura przedmiotu dotyczy w zdecydowanej większości treści w językach innych niż język polski. Problemem z tym związanym jest duża złożoność gramatyki języka polskiego i brak dostępności odpowiednich danych treningowych.
2. Zakres dziedzinowy przeprowadzanych dotąd badań dotyczy głównie normatywnych tekstów prawnych, reguł inżynierii konstrukcji czy biomedycyny. Nie odnaleziono literatury odnoszącej się do dziedziny relacji B2B, co było głównym założeniem projektu. Z tego też powodu np. przyjęte w większości prac ogólne systemy klasyfikacji wyrażeń (zakaz, zobowiązanie, zezwolenie) nie wpisują się w szczegółową dziedzinę projektu.
3. Większość prac podejmuje temat samego aspektu ekstrakcji wyrażeń deontycznych i ich klasyfikacji. Ewaluacja wyrażeń jest poruszana zdecydowanie rzadziej, a jeśli jest to do ewaluacji są wykorzystywane języki i narzędzia formalne (OWL, SVBR, SPARQL etc.), które nie są przyjazne dla użytkowników biznesowych, cyt: “One of the major drawbacks to using these methods is that the corpus for training is not always available,

and businesspeople are not always familiar with (semi) formal models which limits their participation in the validation of generated models”.

4. Prawdą jest, że istniejące badania wykorzystują typowe techniki NLP podobne do wykorzystywanych w projekcie przez Beneficjenta. Wynika to jednak z samej złożoności zagadnienia ekstrakcji i kompilacji reguł. Jest to kolejna warstwa abstrakcji, która jest budowana na bazie istniejących rozwiązań i narzędzi bardziej ogólnej natury, takich jak gotowe algorytmy pakiety NLP, które są wykorzystywane praktycznie w każdej cytowanej pracy naukowej.

Przyjęta metodologia projektowa

Wychodząc od problemu zdefiniowanego we wniosku o dofinansowanie na podstawie przeprowadzonej w pierwszej fazie projektu analizy istniejącego dorobku naukowego i na podstawie własnych przemyśleń zespół projektowy Beneficjenta wyróżnił szczegółowe problemy badawcze składające się na rozwiązanie problemu ekstrakcji wyrażeń deontycznych:

1. P1: Przygotowanie danych treningowych opartych na rzeczywistych dokumentach B2B.
2. P2: Rozpoznawanie w tekście konceptów domenowych (podmiot, obiekt).
3. P3: Rozpoznawanie w tekście rodzaju relacji między konceptami (czasowników, operatorów).
4. P4: Składanie rozpoznanych konceptów i relacji w wyrażenia ewaluowane do wartości prawda lub fałsz.

Badania przemysłowe w projekcie były podzielone na dwa zasadnicze działania:

- Działanie 1: Opracowanie metod przetwarzania tekstu pod kątem ekstrakcji wyrażeń reprezentujących reguły biznesowe.
- Działanie 2: Opracowanie metod translacji reguł biznesowych wyrażonych w tekście na reguły wykonywalne, opracowanie silnika ewaluacji reguł

Działanie 1 dotyczy bezpośrednio problemu badawczego P1 i P2, natomiast Działanie 2 dotyczy problemów P3 i P4. Dla żadnego z wymienionych problemów nie odnaleziono w literaturze gotowych rozwiązań spełniających założone ograniczenia (działanie na skanach dokumentów B2B w języku polskim), i Beneficjent opracował własne podejście do rozwiązania tych problemów. Ze względu na złożoność tych zagadnień nie można uznać, że wykorzystywanie istniejących bibliotek i algorytmów cząstkowych (np. OCR, parsery tekstu itd.) zaprzecza innowacyjności wyników. Analiza literatury naukowej przedmiotu projektu pokazuje, że każde z cytowanych podejść wykorzystuje bazowe techniki NLP, które są zaimplementowane w gotowych bibliotekach. Kluczowym wynikiem prac są algorytmy wyższego poziomu, które realizują postawione cele w kontekście przyjętych dla projektu założeń i ograniczeń biznesowych.

Problem badawczy numer P1 obejmował przygotowanie danych treningowych specyficznych dla przyjętych ograniczeń projektowych. Należy zauważyć, że już samo to zagadnienie było wyzwaniem nie mającym gotowego rozwiązania gdyż nie istnieją w ogóle ogólnie dostępne

zbiory danych takich dokumentów, które spełniałyby założenia projektu. Warto zauważyć, że wśród cytowanych badań przeważa wykorzystanie dokumentów w języku angielskim, ew. chińskim i hiszpańskim. Ponadto cytowane prace badawcze są realizowane na zbiorach dokumentów, które są publikowane w formacie dokumentów natywnie cyfrowych (xml i inne formaty tekstowe) dla których nie występuje problem ekstrakcji tekstu z obrazu. Zakres tematyczny tych dokumentów dotyczy zaś obszarów prawodawstwa krajowego, np. prawa pracy czy reguł prawa budowlanego. W opinii Beneficjenta w związku z tym kluczowe było rozwiązanie szczegółowych problemów:

1. Pozyskanie reprezentatywnego zbioru danych.
2. Przetworzenie uzyskanego zbioru danych umożliwiające operowanie na tekście technikami NLP.
3. Annotowanie tekstu pod kątem występowania w tekście istotnych dla projektu danych.

Punkt 1 Beneficjent mógł zrealizować dzięki swemu doświadczeniu w branży i posiadanym relacjom z klientami, dzięki którym udało się pozyskać odpowiedni dostęp i zezwolenia na wykorzystanie danych.

Punkt 2 obejmował prace, które wykorzystywały techniki przetwarzania jak OCR i podstawowe przetwarzanie NLP (tokenizacja, lematyzacja itd.). Wykonanie tego kroku było zasobochłonne i pracochłonne, ale jednocześnie niezbędne i konieczne do realizacji dalszych prac. Wykorzystanie istniejących bibliotek i algorytmów do takich zadań jest przy dzisiejszym stanie wiedzy oczywiste. Należy jednak pamiętać, że te narzędzia muszą być odpowiednio sparametryzowane i konieczne są eksperymenty związane z doбором tych parametrów i takie eksperymenty zostały przeprowadzone. Ze względu na specyfikę danych wejściowych (skany dokumentów) oraz niedoskonałość narzędzi OCR konieczne było opracowanie metod wektoryzacji, które pomimo błędów w danych pozwalały skutecznie stosować algorytmy wyższego poziomu (np. klasyfikatory).

W punkcie 3 dotyczącym adnotacji tekstów Beneficjent zastosował w pełni innowacyjne podejście do annotacji danych treningowych. Typowe podejście wykorzystywane w cytowanych pracach zakłada ręczne annotowanie danych treningowych. Realizowane jest to poprzez zaznaczanie w tekście wystąpień interesujących danych przez ludzi. Beneficjent w toku projektu przygotował również narzędzie informatyczne wraz z interfejsem użytkownika umożliwiające ręczne annotowanie danych. Jednak ze względu na ilość danych byłby to proces bardzo czasochłonny i podatny na błędy. Ze względu na specyfikę działalności Beneficjent posiada dostęp do zawartości metadanych powiązanych z dokumentami w bazie danych. Wykorzystanie tych metadanych nie rozwiązuje jednak problemu annotacji, gdyż oprócz znajomości wartości metadanych konieczne jest prawidłowe umiejscowienie jej wystąpienia w tekście dokumentu. Aby to skutecznie zrobić należy uwzględnić kilka specyficznych problemów:

1. Błędy literowe w treści tekstu czytanego z dokumentów przez OCR.
2. Błędy w treści metadanych wyniku błędów użytkownika podczas wprowadzania danych.
3. Różnice w formacie danych w metadanych i w treści dokumentów (np. różny format dat czy zamiana kolejności tokenów np imię nazwisko vs nazwisko imię)

Występowanie tych problemów poskutkowało opracowaniem dedykowanego algorytmu dopasowywania danych, który uwzględnia oprócz bezpośredniego porównywania tokenów tekstowych również dopasowanie rozmyte na podstawie odległości levensteina oraz podejście

kombinatoryczne zakładające porównywanie zapisów wartości danych w różnych formatach oraz bi-gramy pozwalające uniezależnić algorytm od kolejności wystąpień tokenów. **Takie podejście jest w pełni innowacyjną oraz autorską techniką zastosowaną przez Beneficjenta i jest nową wiedzą uzyskana w projekcie.**

W przekonaniu Beneficjenta i w kontekście przebadanej literatury tematu problemy P2 i P3, czyli “Rozpoznawanie w tekście konceptów domenowych (podmiot, obiekt)” oraz “Rozpoznawanie w tekście rodzaju relacji między konceptami (czasowników, operatorów)”, są problemami klasyfikacji. W obu przypadkach mamy do czynienia z ustalonym z góry systemem klasyfikacji, który jest specyficzny dla danej dziedziny i oparty na z góry zdefiniowanej ontologii, czyli na znanych rodzajach obiektów i operatorów definiujących relacje B2B pomiędzy tymi obiektami. Takie podejście jest również rekomendowane w większości cytowanych prac. W dziedzinie rozważanej przez Beneficjenta uznano, że przyjęte obiekty i ich atrybuty są zdefiniowane przez strukturę rodzajów klas i atrybutów dokumentów zdefiniowanych w systemie obiegu dokumentów. Z kolei wartości atrybutów tych dokumentów reprezentują proste typy danych reprezentujące wartości lub obiekty świata rzeczywistego (liczby, daty, kontrahentów i ich dane charakterystyczne). Taka natura wartości atrybutów narzuca z kolei interpretację dostępnych operatorów (czasowników). Są to operatory porównywania danych (podobnie zresztą jak w cytowanym artykule dot. reguł inżynierii konstrukcji budowlanych).

Ze względu na kluczowe wskazane czynniki projektowe (aplikacja rozwiązania do języka polskiego i związany z nimi brak danych) konieczne było opracowanie podejścia, które będzie skuteczne wobec tych ograniczeń. Proponowane w większości prac podejście regułowe ma tę dobrą cechę, że nie wymaga dużej ilości danych. Jednak próby zastosowania takiego podejścia do języka polskiego skutkowało dużą złożonością i różnorodnością szablonów reguł gramatycznych czyniąc to podejście nie praktycznym ze względu na ogromną pracochłonność przy tworzeniu takiego zbioru reguł, który wyczerpywały różnorodność i złożoność języka prawniczego. W związku z tym należało wyeliminować podejście regułowe. Beneficjent ze względu na prowadzoną działalność posiada dostęp do zbiorów danych dokumentów stosowanych w obrocie B2B, ale jednak ograniczona ilość tych danych (w porównaniu z ilością danych innego typu dostępnych w internecie) ograniczała możliwość trenowania zbyt złożonych modeli np. opartych na sieciach neuronowych (zbyt mała ilość danych w porównaniu do ilości wag do wytrenowania). Dodatkowym problemem była jakość danych. Rzeczywiste dokumenty występujące w obrocie B2B w Polsce to nadal dokumenty papierowe. Dlatego konieczne było wykonanie całego szeregu prac mających na celu przetworzenie posiadanych obrazów dokumentów do postaci możliwej do wykorzystania w procesach NLP. Do właściwego rozwiązania problemu konieczna okazała się właściwa dekompozycja składowych problemu umożliwiającą trenowanie puli współpracujących mniejszych klasyfikatorów zamiast jednego złożonego algorytmu “zintegrowanego”.

Problem numer P4 “Składanie rozpoznanych konceptów i relacji w wyrażenia ewaluowane do wartości prawda lub fałsz” odnosi się do części wykonawczej rozwiązania i zakładał stworzenie algorytmu, który będzie umożliwiał ewaluowanie wyekstrahowanych reguł w kontekście wartości atrybutów konkretnych powiązanych dokumentów. Analizując dotychczasowy dorobek naukowy w większości przypadków jako docelową formę zapisu reguł wybierane były

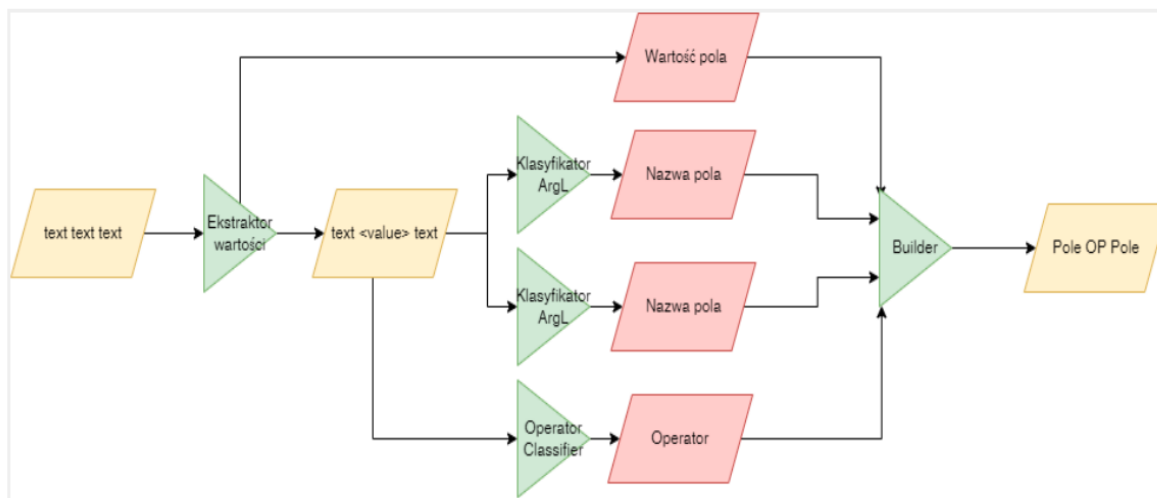
języki formalne, które jednak są trudne do interpretacji dla użytkownika “biznesowego”, co potwierdzają sami autorzy publikacji. Dlatego Beneficjent od początku zakładał stworzenie języka o uproszczonej składni, który będzie bardziej “przyjazny”. Idea konwersji reguł w języku naturalnym na język wykonywalny we wszystkich cytowanych pracach sprowadza się do generowania kodu wykonywalnego na podstawie parametryzacji odpowiednimi obiektami szablonów reguł, które są częścią bazowej ontologii i reprezentują rodzaje relacji deontycznych. W opinii Beneficjenta takie podejście jest prawidłowe z tą różnicą, że wcześniejsze opracowania zakładają oparcie się na istniejących dialektach formalnych w miejscu, gdzie Beneficjent zdecydował się zastosować własny dialekt bardziej dostosowany do kontekstu użycia projektowanego rozwiązania.

Uzyskane rezultaty

W wyniku przeprowadzonych prac wychodząc od dotychczasowego dorobku naukowego w rozważanej dziedzinie udało się opracować algorytm, który jest autorskim rozwiązaniem Beneficjenta i **jest nową wiedzą wytworzoną w projekcie**. Algorytm ten rozwiązuje problem ekstrakcji reguł biznesowych z tekstu w sposób, który jest swego rodzaju odwrotnością podejścia proponowanego w większości cytowanych prac.

Większość dotychczasowych prac bazuje na zestawach heurystycznych reguł opisujących struktury gramatyczne typowe dla wyrażen deontycznych, a następnie identyfikuje obiekty i operatory relacji na podstawie identyfikacji części zdania i zależności między nimi. To, co jest wykonalne dla języka angielskiego jest bardzo trudne dla języka polskiego. Dlatego Beneficjent zaproponował algorytm odwrócony, gdzie najpierw identyfikowane są podstawowe obiekty mogące być przedmiotem relacji deontycznej, następnie tym obiektom nadawana jest odpowiednia rola w zdaniu na podstawie klasyfikacji kontekstu słów otaczających wystąpienie danego obiektu. Ponieważ podstawowymi typami wartości w rozważanej dziedzinie B2B są typy proste reprezentujące takie wartości jak liczby, daty i dane kontrahentów to punktem wyjścia do analizy reguł jest identyfikacja w tekście wystąpień tych podstawowych obiektów. W ten sposób bazowanie na heurystykach dla typowych struktur gramatycznych jest zastąpione uczeniem maszynowym. Właściwe przeprowadzenie procesu normalizacji i wektoryzacji tekstu rozwiązuje też problem bogactwa językowego, w tym synonimów.

Podobnie w przypadku identyfikacji relacji między obiektami brany jest pod uwagę kontekst słów, w którym występują znalezione obiekty i na podstawie tego kontekstu identyfikowany jest rodzaj relacji, przy czym rodzaj ten musi reprezentować jeden z rodzajów relacji występujących we wcześniej przygotowanej ontologii. Tutaj również podejście heurystyczne, które w cytowanych pracach dla języka angielskiego jest zazwyczaj oparte na liście słów i synonimów jest zastąpione uczeniem maszynowym odpowiedniego klasyfikatora. To podejście opracowane przez Beneficjenta w toku prac projektowych daje oczekiwaną skuteczność dla dokumentów B2B w języku polskim i jest to **nowa wiedza wcześniej nie ujawniona w żadnych źródłach**.



Sposób działania algorytmu jest następujący:

1. Tekst jest skanowany przez ekstraktor wartości atrybutów. W przypadku znalezienia atrybutu tekst jest przekazywany dalej w celu określenia właściwego operatora.
2. Klasyfikator statystyczny lub heurystyczny określa rodzaj operatora do zastosowania.
3. KlasyfikatorArgL określa jakie pole jest lewym argumentem wejściowym klasyfikatora.
4. KlasyfikatorArgR określa jakie pole jest prawym argumentem wejściowym klasyfikatora.
5. Następnie składana jest odpowiednia reguła. Wygenerowane w ten sposób reguły mogą być użyte w kroku procesu odpowiedzialnym za walidację krzyżową dokumentów w procesie.

Opisany powyżej algorytm jest innowacyjnym autorskim rozwiązaniem Beneficjenta i jest nową wiedzą uzyskaną w projekcie.

Składowe opisanego rozwiązania wykorzystują istniejące komponenty i biblioteki, ale zestawiają je w nowy sposób wcześniej nie opisany w literaturze przedmiotu, co spełnia warunek innowacyjności. Takie podejście jest zgodne z podstawowymi zasadami inżynierii oprogramowania zakładającymi zasadę "dziel i rządź". Dodatkową cechą opracowanego rozwiązania, która wcześniej nie była zakładana, ale została uzyskana w wyniku prawidłowej dekompozycji problemu jest możliwość stosowania opracowanej metody również do innych kontekstów biznesowych niż zakładane w projekcie przy założeniu posiadania odpowiednich danych. Podejście to składa się z następujących kroków:

1. Pozyskanie dokumentów specyficznych dla danej dziedziny i przetworzenie ich opracowanymi algorytmami przetwarzania dokumentów.
2. Modelowanie ontologii dziedzinowej przez zdefiniowanie obiektów i ich atrybutów za pomocą modelera dokumentów.
3. Modelowanie relacji ontologicznych między dokumentami za pomocą istniejących operatorów lub przez wprowadzenie nowych rodzajów operatorów do języka reguł.

4. Annotowanie danych za pomocą opracowanych narzędzi ręcznie lub przy użyciu algorytmu anotacji zwrotnej jeśli istnieje baza metadanych w formie strukturalnej (bazy danych czy pliku csv, xml lub json).
5. Uruchomienie trenowania klasyfikatorów atrybutów i operatorów na przygotowanym zbiorze danych.
6. Wdrożenie wytrenowanych klasyfikatorów do użycia w procesach biznesowych obejmujących etap ewaluacji dokumentów w opracowanym narzędziu.

Możliwość wykorzystania opracowanych rozwiązań do innych dziedzin niż przewidziane w projekcie świadczy o uniwersalności uzyskanych wyników. Takiej cechy nie posiada żadne z prezentowanych dotąd w literaturze rozwiązań.

Spis literatury

1. Judging Amy: Automated Legal Assessment using OWL 2 Saskia van de Ven, Rinke Hoekstra, Joost Breuker, Lars Wortel, and Abdallah El-Ali 2008
2. Rules from Online Text Saeed Hassanpour, Martin J. O'Connor, Amar Das Stanford Center for Biomedical Informatics Research Stanford, CA, U.S.A. 2011
3. The Semantic of Business Vocabulary and Business Rules: An Automatic Generation From Textual Statements ABDELLATIF HAJ , ABDESSAMD JARRAR , YOUSSEF BALOUKI, AND TAOUFIQ GADIR 2021
4. Automatic Detection and Semantic Formalization of Business Rules Cheikh Kacfar Emani1 2014
5. SBVR based Business Contract and Business Rule IDE Aqueo Kamada Guido Governatori , Shazia Sadiq , 2010
6. Transfer Learning for Deontic Rule Classification: Davide LIGA a,b,1, Monica PALMIRANI University of Luxembourg bAlma Human-AI, University of Bologna 2022
7. Deontic Reasoning for Legal Ontologies Cheikh Kacfar Emani & Yannis Haralambous 2019
8. Agent-Specific Deontic Modality Detection in Legal Language Abhilasha Sancheti, Aparna Garimella, Balaji Vasana Srinivasan, Rachel Rudinger 2022
9. Cognitively Rich Framework to Automate Extraction And Representation of Legal Knowledge, Srishty Saha and Karuna P. Joshi
10. Automated Knowledge Extraction from the Federal Acquisition Regulations System (FARS) Srishty Saha, Karuna P. Joshi, Renee Frank, Michael Aebig, Jiayong Lin
11. Automated Directive Extraction from Policy Texts, Karl Branting, Jim Finegan, David Shin, Stacy Petersen, Alex Lyte, Carlos Balhana, Craig Pfeifer
12. Combining NLP Approaches for Rule Extraction from
13. Legal Documents, Mauro Dragoni, Serena Villata, Williams Rizzi, Guido Governatori
14. On Rule Extraction from Regulations Adam WYNERa and Wim PETERS, 2011

Działanie 1 (badania przemysłowe) - Opracowanie metod przetwarzania tekstu pod kątem ekstrakcji wyrażeń reprezentujących reguły biznesowe

W ramach niniejszego etapu zrealizowano 5 zadań:

1. Przygotowanie narzędzi i infrastruktury do tworzenia i zarządzania zbiorem uczącym.

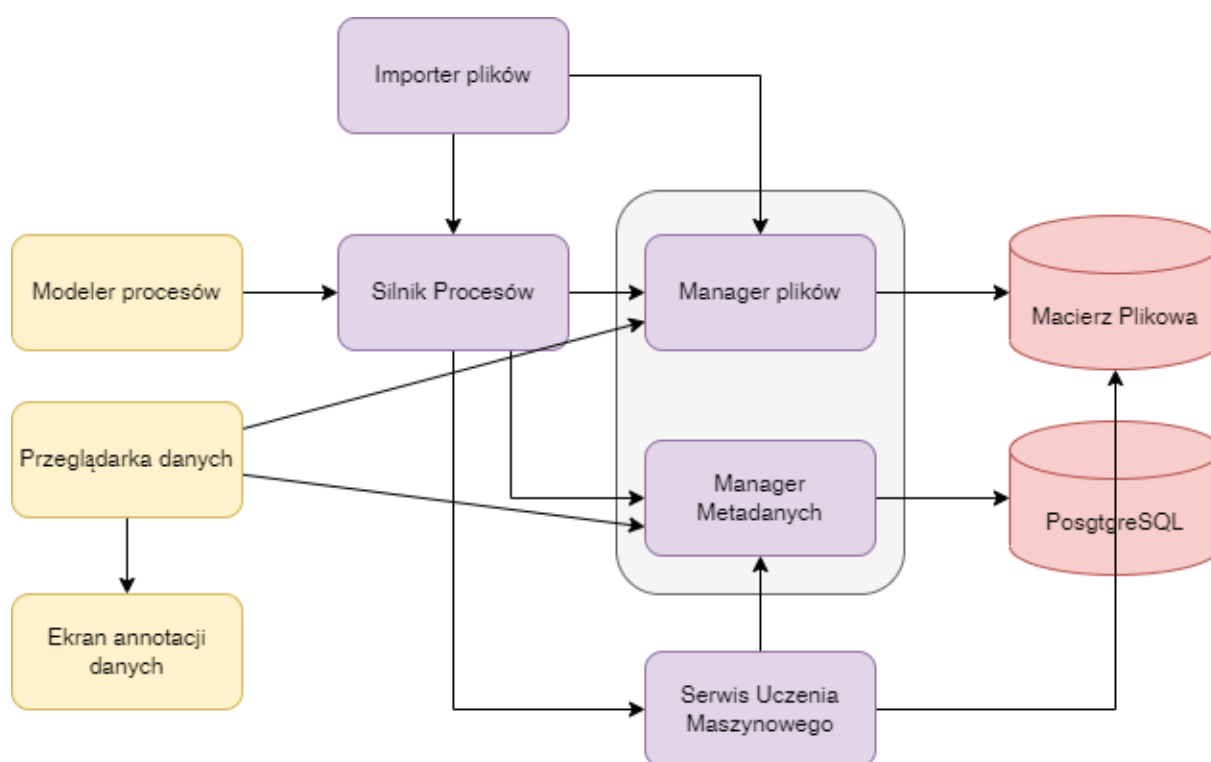
W ramach niniejszego zadania zbudowana została infrastruktura serwerowa do przechowywania i przetwarzania danych z wykorzystaniem GPU oraz narzędzia do adnotowania tekstu i uruchamiania algorytmów przetwarzania tekstu oraz analizy i wizualizacji wyników.

Szczegółowe prace w ramach tego etapu obejmowały:

1. Pozyskanie wymaganej infrastruktury początkowo przez wydzielenie z zasobów własnych oraz przez wynajęcie serwerów wirtualnych.
2. W toku realizacji projektu dokonano zakupów infrastruktury dedykowanej dla projektu. Związane z tym były prace polegające na:
 - a. Instalacji zakupionej infrastruktury w serwerowni.
 - b. Konfiguracji systemowej i sieciowej zainstalowanych serwerów.
 - c. Konfiguracji maszyn wirtualnych w nowej infrastrukturze.
 - d. Przeniesienie na nowe maszyny wirtualne zadań realizowanych wcześniej w chmurze i na istniejących wcześniej serwerach.
3. Wstępna konfiguracja całej infrastruktury obejmowała:
 - a. Instalację komponentów środowiska Proxmox umożliwiającego zarządzanie maszynami wirtualnymi.
 - b. Instalację środowiska Salt umożliwiającego automatyzację dostarczania wymaganych zasobów obliczeniowych w ramach postępu projektu i bieżących wymagań projektowych.
 - c. Konfigurację środowiska docker i narzędzia docker compose, integracja ze środowiskiem jenkins wykorzystywanym jako narzędzie do budowania i dostarczania komponentów.
 - d. Instalacja środowiska OpenVPN i IPA umożliwiającego zdalny dostęp i korzystanie z serwerów dla zespołu projektowego.
 - e. Instalację środowiska zabix umożliwiającego monitoring działania serwerów.
 - f. Konfigurację maszyn wirtualnych w zależności od roli:
 - i. obsługa repozytorium plików
 - ii. serwerów do prowadzenia badań (linux, python, jupyter notebook, wymagane biblioteki do przetwarzania NLP)
 - iii. serwerów do prototypowania rozwiązania końcowego (linux, postgresql, java, wymagane biblioteki)
4. W toku całego projektu wykonywane były bieżące prace administracyjne i typu devops związane z koniecznością konfigurowania i dostarczania kolejnych elementów środowiska oraz nadzoru nad działaniem tego środowiska. Prace te angażowały na

bieżący zespół administratorów i programistów w obszarze devops (konfiguracja i weryfikacja rozwiązań i narzędzi programistycznych w środowiskach dostarczanych przez administratorów systemowych).

W ramach tego zadania została przygotowana wymagana infrastruktura, która została oparta na posiadanych przez Beneficjenta zasobach. Opracowana została ogólna architektura rozwiązania i zainstalowane zostały kluczowe komponenty. Ze względu na złożoność procesów przetwarzania dokumentów i ilość spodziewanych danych w projekcie przyjęto, że zasadniczym elementem architektury będzie repozytorium dokumentów wraz z powiązaniem repozytorium metadanych. Przeprowadzono szereg prac programistycznych mających na celu zbudowanie narzędzi potrzebnych do realizacji zadań projektowych. W wyniku tych prac powstała architektura systemu jak na ilustracji.



Poszczególne elementy architektury pełnią następującą rolę:

1. Importer plików - serwis, którego rola jest pobieranie plików z zewnętrznych źródeł, zapisywanie ich w repozytorium plików i inicjowanie procesu przetwarzania plików. Importer umożliwia pobieranie plików z folderów, adresów internetowych oraz ze skrzynek mailowych.
2. Silnik procesów - komponent koordynujący procesy przetwarzania wywołując w odpowiedniej kolejności inne serwisy. Serwisy te realizują konkretne kroki automatycznego przetwarzania danych (plików) lub angażują ludzi np. w kroki związane z anotowaniem lub weryfikacją czy uzupełnianiem danych uczących i/lub produkcyjnych.
3. Modeler procesów - narzędzie służące do graficznego modelowania procesów. Procesy te definiują odpowiednie ścieżki przetwarzania danych na etapie pozyskiwania i przetwarzania danych uczących oraz pozwala modelować produkcyjne procesy przetwarzania dokumentów z wykorzystaniem wytworzonych modeli ML.

4. Manager plików - serwis zarządzający repozytorium plików odpowiedzialny za nadawanie plikom unikalnych identyfikatorów, składowanie plików w macierzy, generowanie miniatur itd.
5. Manager metadanych - serwis zarządzający modelowaniem i składowaniem metadanych powiązanych z plikami. Każdy rodzaj pliku może posiadać różnego rodzaju meta dane (atrybuty), które są uzupełniane na etapie przetwarzania plików. Serwis umożliwia przechowywanie, wyświetlanie i wyszukiwanie powiązanych plików dokumentów na podstawie metadanych.
6. Przeglądarka danych - serwis zawierający interfejs użytkownika umożliwiający wyszukiwanie i przeglądanie danych zgromadzonych w systemie.
7. Ekran anotacji danych - ekran umożliwiający określenie wartości metadanych dla poszczególnych plików wraz z możliwością określenia miejsca występowania określonych meta danych w treści pliku. Umożliwia również weryfikację i poprawę wartości metadanych odczytanych przez algorytmy ML.
8. Serwis uczenia maszynowego - serwis składający się z wielu podserwisów realizujących funkcje wywoływania i trenowania modeli ML. Serwisy te implementowane są w języku python i wykorzystują szereg bibliotek tego języka dedykowanych do realizacji zadań ML. Głównym komponentem tego serwisu jest biblioteka SpaCy ale istnieje też możliwość instalowania innych bibliotek. Dzięki architekturze opartej na środowisku docker poszczególne podserwisy mogą wykorzystywać różne wersje bibliotek bez ryzyka konfliktów z możliwością wykorzystania najlepszych dostępnych narzędzi do realizacji danego zadania. Serwis ten ma dostęp do repozytorium plików oraz do powiązanych z plikami metadanych w formacie JSON. Dzięki temu istnieje możliwość uruchamiania procesów trenowania modeli ML przy użyciu danych zgromadzonych w repozytorium i powiązanych z metadanymi. Wytworzone w ten sposób modele ML mogą być następnie publikowane i wykorzystywane w procesach biznesowych, co odbywa się poprzez skonfigurowanie odpowiednich wywołań modeli z poziomu konkretnych procesów biznesowych dla produkcyjnych danych.

Cała architektura została oparta na technologii docker, co umożliwia niezależne wdrażanie i skalowanie poszczególnych serwisów. Separacja serwisów w kontenerach pozwala uniknąć konfliktów konfiguracji i pozwala łączyć serwisy zrealizowane w różnych językach programowania.

W toku całego projektu zostało dostarczonych kilkadziesiąt maszyn wirtualnych w różnych konfiguracjach i do różnych zadań. Ze względu na złożoność projektowanego środowiska (architektura zorientowana na serwisy) zarządzanie i rozwój środowiska był procesem ciągłym trwającym przez cały projekt. W trakcie realizacji tych zadań rozwiązano szereg problemów związanych ze złożonością projektowanego rozwiązania, takich jak kompatybilność wersji bibliotek obsługujących GPU z innymi komponentami systemu, co skutkowało koniecznością granulacji systemu na mniejsze serwisy różniące się konfiguracją, ale dedykowane do realizacji wąsko określonych zadań. **Zakres wykonanych prac w ramach tego zadania jest w pełni zgodny z założeniami projektowymi.**

2. Opracowanie i adnotowanie zbioru uczącego.

Podczas realizacji zadania zaplanowano pozyskanie formalnych zezwoleń na wykorzystanie dokumentów biznesowych, zgromadzenie i wstępne porządkowanie danych oraz adnotowanie

i klasyfikację kluczowych fraz w tekście. W ramach realizacji tego zadania wystąpiono o odpowiednie zezwolenia na wykorzystanie danych w projekcie do podmiotów z którymi beneficjent miał nawiązane relacje biznesowe i uzyskano w pierwszej kolejności zgodę na wykorzystanie dokumentów od następujących podmiotów:

- a) Varsovia Capital Sp. z o.o.
- b) Business Online Services Sp. z o.o.
- c) Nature Goods Poland Sp. z o.o.

Ze względu na to, że beneficjent nie miał bezpośredniego wpływu to jakie i kiedy decyzje zostaną podjęte przez kontrahentów i czy będą one pozytywne, oprócz rozpoczęcia procesu formalnego pozyskiwania zgód rozpoczęto niezależnie prace mające na celu umożliwienie prowadzenia przynajmniej części prac niezależnie od uzyskanych zgód na przetwarzanie danych.

W pierwszej kolejności dokonano analizy istniejących w domenie publicznej zbiorów danych z oceną możliwości wykorzystania ich w projekcie. Należy tutaj przypomnieć, że z założeń projektu wynikają bezpośrednio ograniczenia nakładane na zbiór danych wykorzystywanych w projekcie:

- 1. Muszą to być dokumenty w języku polskim ponieważ na ten rynek ma być kierowane docelowe rozwiązanie
- 2. Dokumenty muszą być skanowane i poddawane procesowi OCR, gdyż takie są realia środowisk produkcyjnych. Dokumenty natywnie cyfrowe nie odzwierciedlają realnych wyzwań spotykanych w środowisku produkcyjnym
- 3. Domeną zainteresowania beneficjenta są relacje biznesowe B2B, gdzie jako kluczowe rodzaje dokumentów we wniosku projektowym wskazano faktury i umowy, które są ze sobą powiązane (faktura dotyczy umowy).

Biorąc pod uwagę powyższe wykonano badanie dostępnych rozwiązań w domenie publicznej, **lecz żaden z dostępnych publicznie zbiorów nie spełniał założonych wymagań.** Dlatego beneficjent musiał podjąć decyzje projektowe jak niżej:

- 1. Zdecydowano o wykorzystaniu w pierwszej fazie projektu dokumentów wewnętrznych należących do beneficjenta i spółek blisko współpracujących z beneficjentem (wskazanych powyżej 3 podmiotów oraz Uniwersyteckiego Centrum Klinicznego).
- 2. Zdecydowano o niezależnym prowadzeniu prac nad procesem digitalizacji dokumentów od NLP.
- 3. W dalszej fazie realizacji projektu po uzyskaniu zgód od klientów posiadających odpowiednie zbiory danych dokonano przejścia z eksperymentami na docelowy zbiór danych.

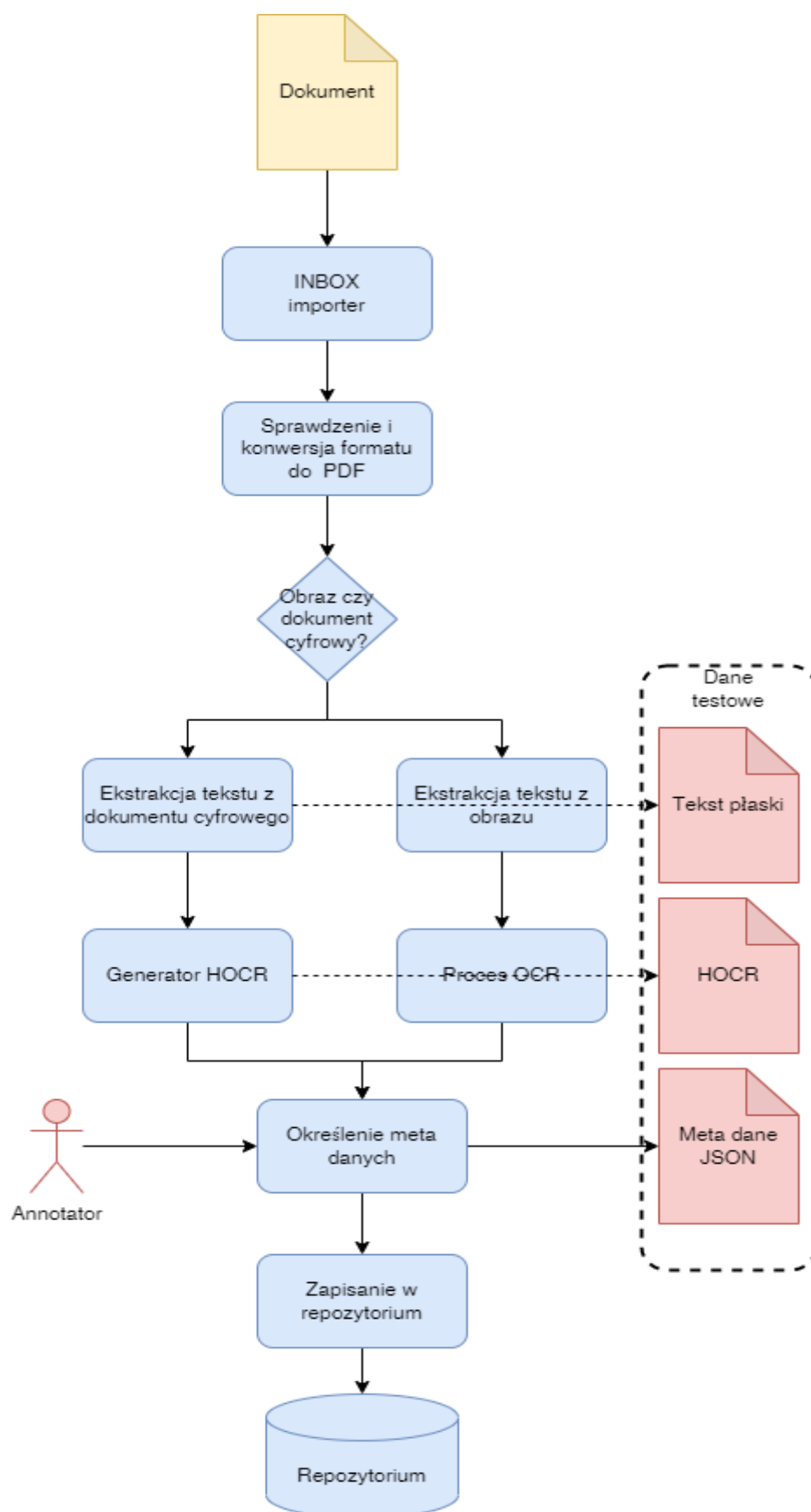
W pierwszej fazie projektu zdecydowano, że eksperymenty będą prowadzone na dokumentach pozyskanych od wyżej wskazanych 3 podmiotów oraz z wewnętrznych zasobów beneficjenta (posiadane umowy i wystawione faktury), w tym na posiadanych danych od Uniwersyteckiego Centrum Klinicznego (dane zostały udostępnione na podstawie zapisów

posiadanych umów o współpracy). Ten zbiór ponad 20 000 dokumentów pozwalał prowadzić prace mające na celu weryfikację podstawowych założeń projektu.

W toku prowadzonych badań nad dostępnym zbiorem dokumentów określono wymagania, jakie musi spełniać docelowy zbiór danych. Są to następujące wymagania:

1. Dokumenty muszą być doprowadzone do jednolitego formatu. Jako format jednolity wybrano format PDF.
2. Tekst treści dokumentu musi być wyekstrahowany do odrębnego pliku dla każdego dokumentu w wersji tzw. plain text (zwykły tekst ułożony w linie)
3. W celu zachowania relacji między poszczególnymi słowami w układzie dokumentów treść dokumentu musi być wyekstrahowana do formatu HOCR, który umożliwia zapamiętanie oprócz samych słów ich pozycji na dokumencie, co okazało się później istotne dla niektórych przypadków ekstrakcji danych.
4. Dla każdego dokumentu zostaną określone kluczowe meta dane, które będą wykorzystywane na dalszych etapach do weryfikacji poprawności ekstrakcji informacji przez porównanie wartości ekstrahowanych w wartościami zapamiętany na tym etapie (tzw. ground truth).

Ze względu na skomplikowanie i dużą złożoność działań, które obejmują również udział człowieka (annotatora) w procesie, proces pozyskiwania i konsolidowania danych został oparty na silniku procesów biznesowych, w którym został zamodelowany proces przetwarzania dokumentów "surowych" do postaci umożliwiającej prowadzenie dalszych prac. Proces ten składa się z szeregu działań automatycznych oraz działań ręcznych z udziałem użytkownika i wygląda w uproszczeniu następująco:



Kolejne kroki procesu obejmują:

1. Import surowych dokumentów z plikowego źródła danych (tzw. folder inbox).
2. Weryfikacja formatu dokumentu i konwersja do formatu docelowego (PDF)
3. Sprawdzenie struktury dokumentu PDF - czy dokument zawiera tylko obraz (skan) w treści dokumentu czy jest to dokument cyfrowy zawierający tekst w treści dokumentu. W zależności od zawartości dokumentu jest on dalej przetwarzany w różny sposób.
4. Ekstrakcja tekstu. Dla dokumentów zawierających obraz konieczne było opracowanie procesu OCR (Optical Character Recognition) w celu wygenerowania tekstu. Jako rozwiązanie tego problemu zostało wybrane narzędzie tesseract. Prowadzony był szereg prac mających na celu dostosowanie parametrów procesu ocr, takich jak sposób segmentacji dokumentu, w celu uzyskania jak najlepszej jakości tego procesu. Jednak pomimo tych prac proces OCR może generować tekst zawierający błędy literowe. Dokładność odzwierciedlenia tekstu jest na poziomie 95% (wartość wyliczona przez porównanie puli tekstu wzorcowego z tekstem, który był drukowany a następnie poddawany procesowi OCR). Błędy procesu OCR, których nie udało się w 100% wyeliminować miały spory wpływ na trudności w dalszych pracach. Dla dokumentów cyfrowych "natywnie" tekst był wyciągany bezpośrednio z treści plików cyfrowych.
5. Ekstrakcja układu tekstu (HOOCR). Dla dokumentów skanowanych poddawanych procesowi OCR wynik w postaci HOOCR został uzyskany przez odpowiednią parametryzację narzędzia tesseract. Dla dokumentów cyfrowych "natywnie" wymagane było opracowanie dedykowanego narzędzia. Narzędzie to na podstawie zawartych w pliku PDF danych opisujących sposób renderowania dokumentów na ekranie określa tzw. bbox (bounding box), w którym występuje dane słowo na wirtualnej "kartce" i na tej podstawie generuje wersję HOOCR treści dokumentu.
6. Określenie wartości metadanych. W tym kroku określane były wartości metadanych, które będą wykorzystywane w dalszym procesie trenowania i weryfikacji algorytmów ekstrakcji. W celu przyspieszenia tego procesu opracowano narzędzie, które umożliwia określanie wartości zdefiniowanych z góry rodzajów (pól) metadanych dla każdego dokumentu wraz z możliwością określenia miejsca (pozycji) wystąpienia danej wartości w treści dokumentu.

W wyniku przeprowadzonych prac uzyskano zbiór danych zawierający ponad 20 tysięcy dokumentów wraz z przypisanymi do dokumentów metadanymi. Poniżej przedstawiono próbki uzyskanego zbioru danych.

	endDate	netValue	installmentPayment	type	signDate	agreementObject				not
0	13-03-2020	33600.00	False	Umowa dostawy	13-03-2018	Dostawa jednorazowych i wielorazowych wyrobów medycznych .				Nał
1	23-02-2020	55400.00	False	Umowa dostawy	23-02-2018	Wyroby medyczne uzywane w zab .urolologicznych .				Nał
2	13-03-2020	327920.00	False	Umowa dostawy	13-03-2018	Dostawa jednorazowych i wielorazowych wyrobow medycznych				Nał
3	21-08-2022	68770.00	False	Umowa dostawy	21-08-2020	Dostawa wyrobów jednorazowych dla UCK				Nał
4	30-09-2023	155218.95	False	Umowa dostawy	30-09-2020	Dostawę odczynników do barwienia dla UCK				Nał
5	02-09-2020	26033.40	False	Umowa dostawy	02-09-2019	Dostawa produktów leczniczych w ramach importu docelowego dla UCK.				Nał
6	31-07-2021	227043.84	False	Umowa dostawy	12-11-2020	Dostawa aparatury medycznej dla UCK				Nał
7	31-07-2021	3879438.61	False	Umowa dostawy	30-11-2020	Dostawę tomografu komputerowego dla UCK				Nał
8	08-03-2023	375700.00	False	Umowa dostawy	08-12-2020	Dostawa wyrobów medycznych do zabiegów koronarografii i koronaroplastyki				Nał
9	08-03-2023	43800.00	False	Umowa dostawy	08-12-2020	Dostawa wyrobów medycznych do zabiegów koronarografii i koronaroplastyki				Nał
	grossValue	zipCode	province	city	street	flat	house	nip	phone	name
	36288.00	02-092	mazowieckie	Warszawa (Włochy)	Zwirki i Wigury	NaN	6 A	5.223086e+09	NaN	Teleflex Polska Sp. z o.o.
	59832.00	75-847	zachodniopomorskie	Koszalin	Wenedów 2	NaN	NaN	6.692256e+09	NaN	Meden-Inmed Sp. z o.o.
	354153.60	64-300	wielkopolskie	Nowy Tomyśl	Tysiąclecia 14	NaN	NaN	7.880009e+09	NaN	Aesculap Chifa Sp. z o.o.
	74271.60	12-200	warmińsko-mazurskie	Pisz	plk. Leona Silickiego	NaN	1	8.490000e+09	NaN	Bialmed Sp. z o.o.
	179558.33	61-626	wielkopolskie	Poznań (Poznań-Stare Miasto)	Szelągowska 30	NaN	NaN	7.781002e+09	NaN	Merck Life Science Sp. z
	28116.07	42-200	dolnośląskie	Częstochowa	Partyzantów 8/10/20	NaN	NaN	7.122995e+09	NaN	Storkpharm Sp. z o.o.
	245207.35	60-118	wielkopolskie	Poznań (Poznań-Grunwald)	Krzywa 13	NaN	NaN	7.831481e+09	NaN	Fresenius Medical Care F
	4216270.50	03-821	mazowieckie	Warszawa (Praga-Południe)	Żupnicza 11	NaN	NaN	1.132886e+09	NaN	Siemens Healthcare Sp. z
	405756.00	31-324	małopolskie	Kraków	Różyckiego 3	NaN	NaN	5.213675e+09	NaN	Aspironix Polska Sp. z o.o.
	47304.00	02-781	mazowieckie	Warszawa (Ursynów)	Rtm. W. Pileckiego 63	NaN	NaN	9.512086e+09	NaN	ProCardia Medical sSp. z
payment			proceedingsNumber	startDate	documentType	id	filename		document text	
	60 dni od otrzymania faktury		156/2017	13-03-2018	Umowy	1076928	1076925-metadata.json		UMOWA DOSTAWY NR 156-9/D	
	60 dni od otrzymania faktury		126/2017	23-02-2018	Umowy	1084036	1084033-metadata.json		UMOWA DOSTAWY NR 126-2/N	
	60 dni od otrzymania faktury		156/2017	13-03-2018	Umowy	1099731	1099728-metadata.json		UMOWA DOSTAWY NR 156-4/D	
	60 DNI OD OTRZYMANIA FAKTURY		84/PN/2020	21-08-2020	Umowy	118433448	118433370-metadata.json		UMOWA DOSTAWY NR 84-2/D-	
	60 dni od otrzymania faktury		178/PN/2020	30-09-2020	Umowy	120306894	120306891-metadata.json		UMOWA DOSTAWY NR 178-1/D	
	60 dni od otrzymania faktury		94/PN/2019	02-09-2019	Umowy	12070640	12070637-metadata.json		UMOWA DOSTAWY NR 94-5/NZ	
	60 DNI OD OTRZYMANIA FAKTURY		137/PN/2020	12-11-2020	Umowy	132386165	132386149-metadata.json		UNIWERSYTECKIE CENTRUM I	
	60 DNI OD OTRZYMANIA FAKTURY		150/PN/2020	30-11-2020	Umowy	136103734	136101797-metadata.json		Fundusze Unia Europejska * * x	
	60 OD OTRZYMANIA FAKTURY		154/PN/2020	08-12-2020	Umowy	144304505	144304493-metadata.json		Ł9287L2000000 UMOWA DOSTA	
	60 DNI OD OTRZYMANIA FAKTURY		154/PN/2020	08-12-2020	Umowy	144635402	144635390-metadata.json		UMOWA DEPOZYTU NR 154-6/I	

Zgromadzone dokumenty poddano adnotacji pod kątem zaznaczenia w treści dokumentów wystąpienia kluczowych encji i fraz, które mają podlegać ekstrakcji i przetwarzaniu. W tym celu dokumenty zostały poddane procesowi digitalizacji. Konieczne w tym celu było wykonanie szeregu prac programistycznych i administracyjnych związanych ze zgromadzeniem, przetworzeniem dokumentów oraz dostarczaniem narzędzi do opisywania (adnotacji) dokumentów.

Na tym etapie realizacji projektu został zidentyfikowany kluczowy problem, który miał wpływ na realizację wszystkich kolejnych zagadnień projektowych. Problem ten polega

na wpływie błędów procesu OCR na działanie wszelkich algorytmów przetwarzania tekstu. W momencie wskazania przez użytkownika wystąpienia poszukiwanych danych w tekście wartość odczytanego tekstu przez mechanizm OCR jest często inna niż tekst określony przez użytkownika jako wartość meta danej. Konieczna była w związku z tym decyzja o niezależnym prowadzeniu prac nad procesem digitalizacji dokumentów. Założenie, że docelowo system ma przetwarzać dokumenty skanowane wymusza wykonywanie na dokumentach procesu OCR a w wyniku przeprowadzonych doświadczeń i porównaniu z wynikami opisanymi w literaturze okazało się, że proces OCR a w szczególności błędy tego procesu mają znaczący wpływ na działanie algorytmów przetwarzania tekstu. W związku z tym konieczne było podjęcie wysiłków mających zminimalizować te błędy.

W tym celu zaprojektowano eksperyment, który zakładał badanie wpływu różnych ustawień i silników OCR na jakość uzyskiwanych danych tekstowych, dodatkowo wprowadzono koncepcję pre-processingu danych na etapie graficznej reprezentacji dokumentu (skanu) metodami przetwarzania obrazu w celu podniesienia jakości graficznej wersji dokumentu poprzez redukcję szumów wpływających na jakość procesu OCR. Zbiór danych obejmował w pierwszej wersji dokumenty wewnętrzne beneficjenta. Dokumenty te zostały zeskanowane i adnotowane (określone zostały podstawowe metadane, "ground truth"):

- dane kontrahenta (nazwa, adres, nip, regon) dla umów i faktur
- dane umowy (numer, data zawarcia, data obowiązywania wartość, termin płatności przedmiot umowy)
- dane faktury (numer, data wystawienia, data sprzedaży, termin płatności, przedmiot faktury).

Eksperyment na tym etapie polegał na tym, że sprawdzano, czy w tekście uzyskanym jako wynik procesu OCR występują tokeny tekstowe, które występują w metadanych. Każdą wartość metadanych skalarnych (data, wartość) potraktowano jako jeden token. Wartość danych tekstowych składających się z wielu słów (nazwa, adres, przedmiot) rozdzielono na tokeny. Do prowadzenia tych prac już na tym etapie była wykorzystywana przede wszystkim biblioteka SpaCy. Testowane były różne silniki OCR: tesseract, easyOCR, paddleOCR. Do oceny jakości procesu OCR w kontekście odczytywania metadanych przyjęto następujący algorytm:

1. Metadane każdego dokumentu podlegają tokenizacji i tworzą zbiór oczekiwanych tokenów o liczebności N-meta.
2. Dokument podlega procesowi OCR i wynik tego procesu podlega tokenizacji.
3. Sprawdzamy dla każdego tokenu z metadanych czy występuje on w spolonizowanej treści dokumentu. Jeśli tak to zwiększamy wartość N-doc reprezentującą ilość odnalezionych tokenów tokenów zgodnych z metadanymi.
4. Finalnie liczymy wartość statystyki tokenów $ST = N\text{-meta} / N\text{-doc}$. Wartość 1 oznacza, że odnaleziono w dokumencie wszystkie tokeny występujące w metadanych określonych jako "ground truth" i tym samym w procesie OCR nie wystąpiły błędy. Wartość mniejsza niż 1 oznacza, że niektóre tokeny reprezentujące metadane nie wystąpiły w treści dokumentu, co świadczy o błędach w treści dokumentu lub innym problemie.

Uzyskane wartości opisanej statystyki, która w praktyce oznacza procent prawidłowo odczytanych z dokumentu tokenów zgodnych z metadanymi były zaskakująco niskie. Kształtowały się na poziomie 0,5-0,7 dla różnych bibliotek OCR, przy czym najlepsze wartości osiągała biblioteka tesseract. Analizując ręcznie dane zauważono, że rzeczywiście w treści tekstu uzyskanego procesie OCR występuje szereg przekłamań: zamienione litery, zaszumienie obrazu odczytywane jako znaki specjalne lub litery (np. i, l itp), wstawianie białych znaków w miejscach, gdzie one nie występują. **Występowanie tego rodzaju błędów OCR znacząco wpływa na działanie algorytmów przetwarzania tekstu i miało ogromny wpływ na dalsze prace projektowe.** Jednocześnie należy podkreślić, że jest to problem istotny z praktycznego punktu widzenia, gdyż z pewnością wystąpi w środowisku produkcyjnym systemu, gdzie jak wspomniano przetwarzane są głównie dokumenty, które nie są natywnie cyfrowe (są skanowane).

Drugi wniosek był związany ze sposobem zapisu wartości metadanych skalarnych, takich jak daty i wartości liczbowe. Część błędów odczytu tych danych wynika z przekłamania przy odczytywaniu podobnych liczb, np. 8 i 9, czy 1 i 7, ale ponadto występuje problem rozpoznawania formatów zapisu i normalizacji wartości przy porównywaniu z metadanymi.

Zespół projektowy podjął szereg działań ukierunkowanych na podniesienie jakości wyników procesu OCR, głównie przez wprowadzenie dwóch dodatkowych etapów przetwarzania dokumentów:

1. Pre processing skanów dokumentów, w celu poprawienia jakości obrazu.
2. Post processing tekstu będącego wynikiem OCR, w celu usunięcia lub korekty błędów i podniesienia statystyki prawidłowo odczytanych tokenów meta danych.

Ekspertyzacja z pre processingiem skanów dokumentów obejmowała różne konfiguracje algorytmów przetwarzania obrazów ukierunkowane na podnoszenie jakości obrazu będącego wejściem dla algorytmu OCR. Wykorzystując narzędzia takie jak biblioteka OpenCV, czy LayoutParser testowano wpływ różnych algorytmów przetwarzania obrazu na statystykę rozpoznawalności tokenów. Zastosowanie biblioteki LayoutParser nie miało pozytywnego wpływu na procent prawidłowo rozpoznawanych tokenów. Natomiast przy wykorzystaniu biblioteki OpenCV na bazie analizy dostępnej literatury związanej z tematem podnoszenia jakości obrazów w wyniku kolejnych iteracji udało się opracować potok przetwarzania składający się z następujących kroków:

System próbował prostować skany za pomocą Tesseract. Algorytm prostowania:

- a) za pomocą tesseract wykonywany był ocr z parametrami zgodnymi z konfiguracją, wynikiem działania plik hocr
- b) poprzez analizę pliku za pomocą wyrażeń regularnych, szukamy parametru:
textangle: textangle ([0-9]+)

```
private Double determineTextangle(final String hocrRaw) {  
    final Matcher m = regexTextangle.matcher(hocrRaw);
```

```
final Map<Double, AtomicInteger> angles = Maps.newHashMap();
int withoutTextangle = StringUtils.countMatches(hocrRaw, OCR_LINE);
while (m.find()) {
    withoutTextangle--;
    angles.computeIfAbsent(Double.parseDouble(m.group(1)), k ->
        new AtomicInteger()).incrementAndGet();
}
angles.put(0.0, new AtomicInteger(withoutTextangle));
return angles.entrySet().stream()
    .sorted((c1, c2) -> Integer.compare(c2.getValue().get(), c1.getValue().get()))
    .map(Map.Entry::getKey)
    .findFirst().orElse(0.0d);
}
```

- c) poprzez analizę pliku za pomocą wyrażeń regularnych, szukamy parametru baseline:
baseline ([-]?[0-9]+([.][0-9]+)?)

```
private Double determineBaselineAngle(final String hocrRaw) {
    final Matcher m = regexBaseline.matcher(hocrRaw);
    final Map<Long, AtomicLong> angles = Maps.newHashMap();
    while (m.find()) {
        final long angleBin = Math.round(Double.parseDouble(m.group(1)) * 1000);
        LongStream.rangeClosed(angleBin - baselineAnglePropagationWindow,
            angleBin + baselineAnglePropagationWindow)
            .forEach(angle -> angles.computeIfAbsent(angle, k -> new AtomicLong())
                .incrementAndGet());
    }
    return angles.entrySet().stream()
        .sorted((c1, c2) -> Long.compare(c2.getValue().get(), c1.getValue().get()))
        .map(Map.Entry::getKey)
        .map(v -> Math.atan((double) v / 1000.0d))
        .map(Math::toDegrees)
        .map(v -> v * -1.0)
        .findFirst().orElse(0.0d);
}
```

- d) dla odnalezionych kątów, wykonywany jest obrót, jeżeli obliczony kąt obrotu jest większy niż zadany próg

Dodatkowe procesowanie w celu zwiększenia jakości wykonuje skrypty

Usuwanie linii: (istotne dla tabel w fakturach, np. tabelka vat, usunięcie linii spowodowało polepszenie jakości odczytywania liczb w takich scenariuszach)

```
import cv2
import getopt
import numpy as np
import sys

def remove_lines_using_morphology(img, tresh, scale=30):
    horizontal = tresh.copy()
    vertical = tresh.copy()

    height, width, channels = img.shape
    # play with this variable in order to increase / decrease the amount of lines to be detected

    # Specify size on horizontal axis int
    horizontalsize = int(width / scale)

    horizontalStructure = cv2.getStructuringElement(cv2.MORPH_RECT, (horizontalsize, 1))

    horizontal = cv2.erode(horizontal, horizontalStructure, iterations=-1)
    horizontal = cv2.dilate(horizontal, horizontalStructure, iterations=-1)

    verticalsize = int(height / scale)

    verticalStructure = cv2.getStructuringElement(cv2.MORPH_RECT, (1, verticalsize))

    vertical = cv2.erode(vertical, verticalStructure, iterations=-1)
    vertical = cv2.dilate(vertical, verticalStructure, iterations=-1)

    mask = cv2.cvtColor(horizontal + vertical, cv2.COLOR_BGR2GRAY)
    mask = cv2.dilate(mask, np.ones((3, 3)), iterations=1)

    return cv2.cvtColor(mask, cv2.COLOR_GRAY2BGR)

def skeleton(img):
    sought = 255
    result = np.count_nonzero(np.all(img != sought, axis=0))

    if result != 0:

        size = np.size(img)
        skel = np.zeros(img.shape, np.uint8)
```

```
ret, img = cv2.threshold(img, 127, 255, 0)
element = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))
done = False

while (not done):
    eroded = cv2.erode(img, element)
    temp = cv2.dilate(eroded, element)
    temp = cv2.subtract(img, temp)
    skel = cv2.bitwise_or(skel, temp)
    img = eroded.copy()

    zeros = size - cv2.countNonZero(img)
    if zeros == size:
        done = True
    skel = cv2.dilate(skel, np.ones((2, 2)))

    return skel
else:
    return img

def remove_lines(img, is_skeleton):
    morph = clarity(img)

    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (1, 1))
    morph = cv2.morphologyEx(morph, cv2.MORPH_CLOSE, kernel)
    morph = cv2.morphologyEx(morph, cv2.MORPH_OPEN, kernel)

    # split the gradient image into channels
    image_channels = np.split(np.asarray(morph), 3, axis=2)

    channel_height, channel_width, _ = image_channels[0].shape

    # apply Otsu threshold to each channel
    for i in range(0, 3):
        _, image_channels[i] = cv2.threshold(~image_channels[i], 0, 255, cv2.THRESH_OTSU |
cv2.THRESH_BINARY)
        image_channels[i] = np.reshape(image_channels[i], newshape=(channel_height,
channel_width, 1))

    # merge the channels
    image_channels = np.concatenate((image_channels[0], image_channels[1],
image_channels[2]), axis=2)
```

```
ret, thresh = cv2.threshold(cv2.cvtColor(image_channels, cv2.COLOR_BGR2GRAY), 240,
255, cv2.THRESH_BINARY)

if is_skeleton:
    thresh = skeleton(thresh)

thresh = cv2.cvtColor(thresh, cv2.COLOR_GRAY2BGR)

mask = cv2.cvtColor(remove_lines_using_morphology(img, thresh),
cv2.COLOR_BGR2GRAY)

dst = cv2.inpaint(img, mask, 3, cv2.INPAINT_TELEA)

return dst

def decrease_brightness(img, value):
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    # decreasing the V channel by a factor from the original
    hsv[..., 2] = hsv[..., 2] * value

    img = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)

    return img

def increase_brightness(img, value=20):
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    h, s, v = cv2.split(hsv)

    lim = 255 - value
    v[v > lim] = 255
    v[v <= lim] += value

    final_hsv = cv2.merge((h, s, v))
    img = cv2.cvtColor(final_hsv, cv2.COLOR_HSV2BGR)

    return img

def clarity(img):
    hist = cv2.calcHist([img], [0], None, [256], [0, 256])

    (minVal, maxVal, minLoc, maxLoc) = cv2.minMaxLoc(hist)
    if maxLoc[1] < 240:
```

```
img = img
# For tests: does nothing
# img = increase_brightness(img, 255-maxLoc[1])
# img = cv2.addWeighted(img, 1.1, img, 0, 0)
else:
    img = cv2.addWeighted(img, 1.6, img, 0.1, -300)
return img

def main(argv):
    skeleton = False
    help = 'removeLines.py -s <skeleton>'
    try:
        opts, args = getopt.getopt(argv, "h:s:", ["skeleton="])
    except getopt.GetoptError:
        print(help)
        sys.exit(2)

    if len(opts) == 0:
        print(help)
        sys.exit(2)

    for opt, arg in opts:
        if opt == '-h':
            print(help)
            sys.exit()
        elif opt in ("-s", "--skeleton"):
            skeleton = bool(arg)
        else:
            print(help)
            sys.exit()

    stdin = sys.stdin.buffer.read()
    array = np.frombuffer(stdin, dtype='uint8')
    img = cv2.imdecode(array, 1)

    dst = remove_lines(img, skeleton)

    sys.stdout.buffer.write(cv2.imencode('.png', dst)[1].tostring())

if __name__ == "__main__":
    main(sys.argv[1:])
```

Poprawa jasności (na podstawie analizy histogramu, dla dokumentów szarych)

```
import cv2
import numpy as np
import sys

def increase_brightness(img, value=20):
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    h, s, v = cv2.split(hsv)

    lim = 255 - value
    v[v > lim] = 255
    v[v <= lim] += value

    final_hsv = cv2.merge((h, s, v))
    img = cv2.cvtColor(final_hsv, cv2.COLOR_HSV2BGR)

    return img

def clarity(img):
    hist = cv2.calcHist([img], [0], None, [256], [0, 256])

    (minVal, maxVal, minLoc, maxLoc) = cv2.minMaxLoc(hist)
    if maxLoc[1] < 250:
        img = increase_brightness(img, 250 - maxLoc[1])
        img = cv2.addWeighted(img, 1.1, img, 0, 0)

    result = cv2.resize(img, (0, 0), fx=2, fy=2)
    return result

def main(argv):
    stdin = sys.stdin.buffer.read()
    array = np.frombuffer(stdin, dtype='uint8')
    img = cv2.imdecode(array, 1)

    dst = clarity(img)

    sys.stdout.buffer.write(cv2.imencode('.png', dst)[1].tostring())

if __name__ == "__main__":
    main(sys.argv[1:])
```


Usuwanie odizolowanych pojedynczych i podwójnych ciemnych pikseli.

```
import cv2
import bosimage
import sys
import numpy as np

def eat_breadcrumbs(img, max_size=2):
    bg = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    _, bg = cv2.threshold(bg, 240, 255, cv2.THRESH_BINARY)
    bosimage.filter_regions(img, bg, 127, max_size+1)
    return img

def main(argv):
    stdin = sys.stdin.buffer.read()
    array = np.frombuffer(stdin, dtype='uint8')
    img = cv2.imdecode(array, 1)
    dst = eat_breadcrumbs(img)
    sys.stdout.buffer.write(cv2.imencode('.png', dst)[1].tostring())

if __name__ == "__main__":
    main(sys.argv[1:])
```

```
cpdef unsigned char[:, :] fill_vertical_gaps(unsigned char[:, :] image, int size, unsigned char
color):
    cdef int x, y, w, h, z, n
    cdef unsigned char current
    h = image.shape[0]
    w = image.shape[1]
    for y in range(0, h):
        last = 0
        x = 0
        while x < w:
            current = image[y, x]
            if current == 255 - color and last == color:
                z = x + 1
                while z < w and image[y, z] == 255 - color:
                    z+=1
```

```

        if z - x < size:
            for n in range(x, z):
                image[y, n] = color
            x = z
        x += 1
        last = current
    return image

cpdef unsigned char[:, :] fill_horizontal_gaps(unsigned char[:, :] image, int size, unsigned
char color):
    cdef int x, y, w, h, z, n
    cdef unsigned char current
    h = image.shape[0]
    w = image.shape[1]
    for x in range(0, w):
        last = 0
        y = 0
        while y < h:
            current = image[y, x]
            if current == 255 - color and last == color:
                z = y + 1
                while z < h and image[z, x] == 255 - color:
                    z += 1
                if z - y < size:
                    for n in range(y, z):
                        image[n, x] = color
                y = z
            y += 1
            last = current
    return image

cpdef int count_black(unsigned char[:, :] image, int x1, int y1, int x2, int y2):
    cdef int x, y, c
    c = 0
    for y in range(y1, y2):
        for x in range(x1, x2):
            c += 1 if image[y, x] == 0 else 0
    return c

cpdef int[:] create_pixel_histogram(unsigned char[:, :] image):
    cdef int x, y, pixel
    cdef int[256] pixels = [0] * 256

```

```

for y in range(0, image.shape[0]):
    for x in range(0, image.shape[1]):
        pixel = image[y, x]
        pixels[pixel] = pixels[pixel] + 1
return pixels

cdef bint is_black_or_empty(unsigned char[:, :] image, int x, int y, int th):
cdef int height = image.shape[0]
cdef int width = image.shape[1]
return 0 <= x < width and 0 <= y < height and image[y, x] < th

def walk_black(unsigned char[:, :] image, int x, int y, int th):
if not is_black_or_empty(image, x, y, th):
    return []
points = [(x, y), (x - 1, y), (x + 1, y), (x, y - 1), (x, y + 1)]
return [point for point in points if is_black_or_empty(image, point[0], point[1], th)]

def walk_black_rec(unsigned char[:, :] image, int x, int y, int th):
to_visit = walk_black(image, x, y, th)
visited = set([])
while to_visit:
    point = to_visit.pop()
    visited.add(point)
    walked = walk_black(image, point[0], point[1], th)
    to_visit.extend([p for p in walked if p not in visited])
    visited.update(walked)
return list(visited)

def find_regions(unsigned char[:, :] image, int th):
cdef int x, y
cdef int height = image.shape[0]
cdef int width = image.shape[1]
regions = []
visited = set()
for y in range(0, height):
    for x in range(0, width):
        if (x, y) not in visited:
            points = walk_black_rec(image, x, y, th)
            if len(points) > 0:
                regions.append(points)
                visited.update(points)

```

```
return regions

cpdef void filter_regions(unsigned char[:, :, :] image, unsigned char[:, :] imgray, int th, int
size):
    cdef int x, y
    regions = find_regions(imgray, th)
    for region in regions:
        if len(region) < size:
            for x, y in region:
                image[y, x, 0] = 255
                image[y, x, 1] = 255
                image[y, x, 2] = 255
```

Wykonanie OCR

Do wykonywania OCR służy tesseract uruchamiany z konfiguracją:

Wersja: tesseract 4.1.1

leptonica-1.80.0

libgif 5.2.1 : libjpeg 8d (libjpeg-turbo 2.0.6) : libpng 1.6.37 : libtiff 4.2.0 : zlib 1.2.11 : libwebp
1.1.0

Found AVX2

Found AVX

Found FMA

Found SSE

Język zgodnie z konfiguracją: pol

Tryb segmentacji: SPARSE_TEXT(11),

Wynikiem działania jest plik hocr. Który następnie może podlegać dalszej analizie (np. określenie typów danych: NIP, liczba, data itd).

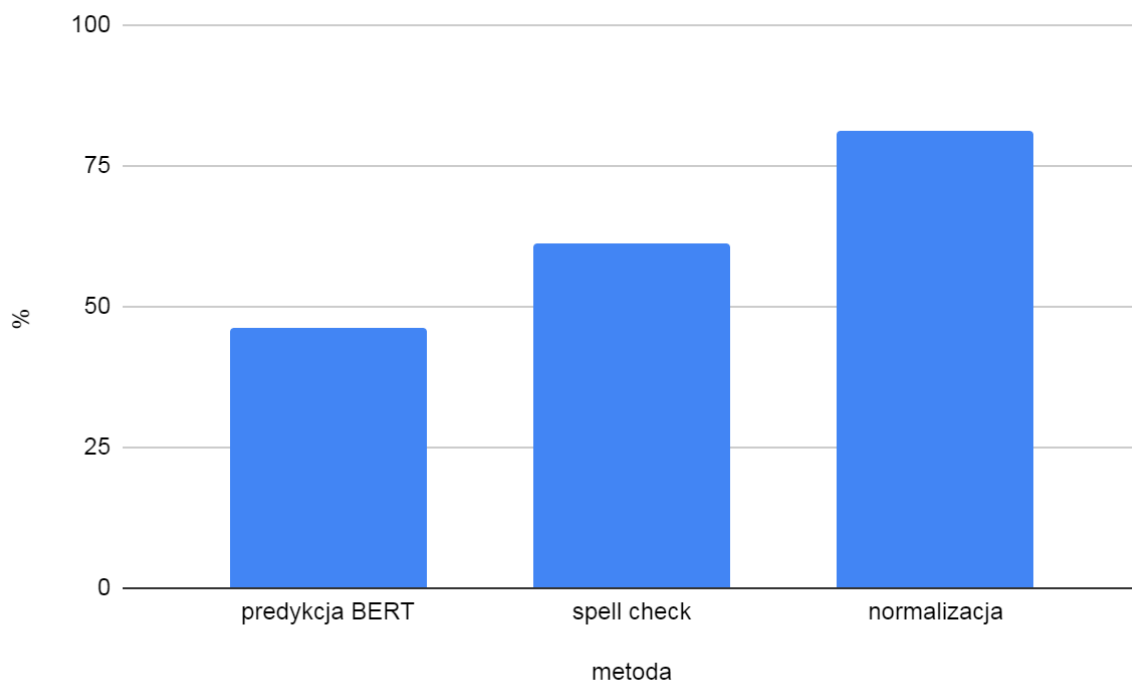
Powyższe rozwiązanie obejmuje wiele kroków i parametrów, które były heurystycznie dobierane w kolejnych iteracjach. Metodą oceny wpływu tych parametrów na wynik była wcześniej opisana statystyka, która określała jaki procent z tokenów występujących w metadanych został odnaleziony w treści tekstu będącego wynikiem procesu OCR.

W wyniku zastosowania pre processingu skanów udało się podnieść procent prawidłowo rozpoznawanych tokenów metadanych do wartości 0,8. Oznaczało to jednak, że nadal duża część tokenów (20%), których wystąpienie było oczekiwane ze względu na obecność w metadanych nie była rozpoznawana. **Już na tym etapie projektu widoczne było, że jakość danych wejściowych (skanów) oraz jakość uzyskanego tekstu jest daleka od ideału i będzie miała ogromny wpływ na wyniki.**

Na tym etapie zdecydowano o konieczności wprowadzenia dodatkowego etapu post processingu tekstu, który był wynikiem OCR w celu podniesienia jego jakości. Zdecydowano o wykorzystaniu algorytmów, które bazują na wytrenowanych modelach tekstu i na podstawie statystycznych modeli dążą do korekty błędnych słów występujących w tekście. Do testów zakwalifikowano następujące algorytmy:

1. Contextual Spell Check wykorzystujący model BERT do prognozowania słowa na podstawie kontekstu jego wystąpienia. Błędne słowo ma być zastąpione słowem poprawnym na podstawie otaczającego kontekstu.
2. Algorytm autocorrect bazujący na klasycznym algorytmie spell chcel Petera Norviga. Algorytm wykorzystuje odległość Levenshteina między słowami na bazie znanej listy słownikowej i zamienia błędne słowo na najbardziej zbliżone słowo ze słownika.
3. Własny algorytm normalizacji wykorzystujący podstawowe operacje na tekście (normalizacja słów, usuwanie znaków niepożądanych za pomocą wyrażeń regularnych, usuwanie tokenów uznanych za niepożądane na podstawie modelu statystycznego pl_core_news).

metoda	%
predykcja BERT	46,34
spell check	61,41
normalizacja	81,41



Normalizacja danych tekstowych

Zbiór danych został podzielony na zbiór treningowy (80% dokumentów) i zbiór testowy (20% dokumentów). Testowane są następujące sposoby przetwarzania dokumentów:

Filtracja i normalizacja tokenów

Przy użyciu biblioteki SpaCy i pretrenowanego modelu „pl_core_news_lg” treść dokumentów została zanotowana.

Wczytanie modelu SpaCy

```
import spacy

nlp_model = spacy.load("pl_core_news_lg")
```

W ten sposób tekst podzielony został na słowa/znaki (tokeny) – każde z odpowiadającymi mu anotacjami/tagami. Przed podziałem na tokeny z tekstu dokumentu usunięte zostały znaki interpunkcyjne oraz wielokrotne spacje.

```
document_part_text = "sekretarz sądowy Agnieszka L. Pismo wygenerowane elektronicznie. Pismo nie wymaga podpisu własnoręcznego na podstawie 6 : 19 ust.4 zarządzenia Ministra Sprawiedliwości z dnia 12 grudnia 2003 r. w sprawie organizacji i zakresu działania sekretariatów sądowych oraz innych działów administracji sądowej jako właściwe zatwierdzone w sądowym systemie teleinformatycznym."
```

```
import re

PUNCTUATION = "!\"#$%&'*+,-./:;=?@[\\]^_`{|}~",""

def remove_punctuation(text):
    return text.translate(str.maketrans("", "", PUNCTUATION))

def remove_multiple_spaces(text):
    return re.sub(" +", " ", text)

document_part_text = remove_punctuation(document_part_text)
document_part_text = remove_multiple_spaces(document_part_text)
document_tokens = nlp_model(document_part_text)

for token in document_tokens:
    print(token)
```


sekretny
sądowy
Agnieszka
L
Pismo
wygenerowane
elektronicznie
Pismo
nie
wymaga
podpisu
własnoręcznego
na
podstawie
6
19
ust4
zarządzenia
Ministra
Sprawiedliwości
z
dnia
2
grudnia
2003
r
w
sprawie
organizacji
i
zakresu
działania
sekreteriów
sądowych
oraz
innych
działów
administracji
sądowej
jako
właściwe
zatwierdzone
w
sądowym
systemie
teleinformatycznym

Z biblioteki SpaCy pobrana została lista tokenów, które uznawane są za nieistotne przy rozwiązywaniu problemów przetwarzania języka naturalnego, takich jak klasyfikacja dokumentów.

```
import os

data_path = os.path.join(get_workspace_dir(), "data")
with open(os.path.join(data_path, 'stopwords_pl'), 'r') as f:
    stopwords = f.readlines()
with open(os.path.join(data_path, 'stopwords_pl_ext'), 'r') as f:
    stopwords += f.readlines()

stopwords = [word.lower().strip() for word in stopwords]
for word in ["pan", "pana", "pani"]:
    stopwords.remove(word)
```

Lista ta zawiera 381 znormalizowanych słów, tak zwanych stopwords, przykładowo:

"cała", "dla", "przy", "sposob", "wielu", "o", "ci", "niemu", "ktora",
"jakichś", "tel", "kto", "xv", "nasz", "niego", "każdy", "powinni",
"prawie", "ja", "godz", "czyli", "to", "mną", "dosc", "bedzie", "zaś", "xii",
"dwaj", "jeden", "moje", "totez", "ona", "której", "aby", "gdyz",
"rowniez", "żadne", "bynajmniej", "mna", "oni", "tylko", "mamy",
"kierunku"

Normalizacja tokenów:

- Tokeny, które są rozpoznawane przez model SpaCy NER jako encje typu persName, date lub orgName, zamienione zostały na symbole zastępcze odpowiadające ich typom.
- Symbole zastępcze występujące jeden po drugim połączone zostały w jeden.
- Tokeny, które są na liście stopwords, zostały usunięte.
- Pozostałe tokeny zastąpione zostały przez ich znormalizowaną formę (lemma). W tej formie zostały wybrane jako słowa kluczowe dokumentów.

```
ENTITY_TYPES = ['persName', 'date', 'orgName']

def normalise_single_token(token, prev_ent_type, stopwords):
    lemma = token.lemma_.lower()
    ent_type = token.ent_type_

    if ent_type in ENTITY_TYPES:
        if ent_type != prev_ent_type:
```

```

        return ent_type, f"<{ent_type}>"

    elif lemma not in stopwords:
        return "", lemma

    return prev_ent_type, ""

def normalise_tokens(tokens, stopwords):
    prev_ent_type = ""
    normalised_token_list = []
    for token in tokens:
        prev_ent_type, normalised_token = normalise_single_token(
            token=token, prev_ent_type=prev_ent_type,
            stopwords=stopwords)
        if normalised_token:
            normalised_token_list.append(normalised_token)

    return normalised_token_list

def list_to_string(list_to_join, connector=" "):
    return connector.join(list_to_join)

def preprocess(tokens):
    tokens = normalise_tokens(tokens=tokens, stopwords=stopwords)
    return list_to_string(tokens, connector=" ")

preprocessed_doc = preprocess(document_tokens)

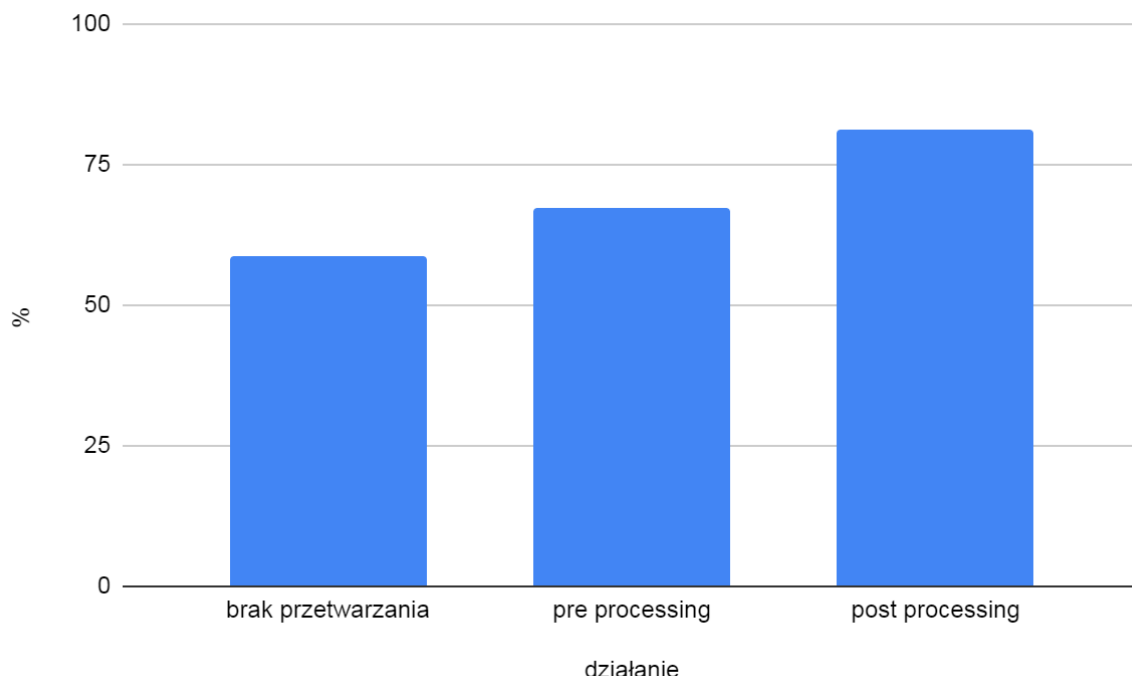
print(preprocessed_doc)

```

sekretarz sądowy <persName> pismo wygenerować elektronicznie pismo
wymagać podpis własnoręczny podstawa 6 19 ust4 zarządzenie ministra
sprawiedliwość <date> sprawa organizacja zakres działanie sekretariat
sądowy dział administracja sądowy właściwy zatwierdzić sądowy system
teleinformatyczny

operacja	%
brak przetwarzania	58,63
pre processing	67,38

post processing	81,41
-----------------	-------



Dalsza analiza pokazała, że problem na tym etapie dotyczył już w większym stopniu formatowania danych.

Ze względu na rozłożenie w czasie procesu pozyskiwania dokumentów od współpracujących klientów prace nad zbiorem danych prowadzone były de facto iteracyjnie przez cały projekt. **Ostateczny zbiór danych obejmuje 2357 powiązanych par dokumentów (faktur i umów) reprezentatywnych dla rozwiązywanego problemu, które pochodziły z rzeczywistego środowiska (dokumenty te zostały wyselekcjonowane z pierwotnego zbioru ponad 20 tysięcy dokumentów).**

3. Ewaluacja i wybór podstawowych bibliotek przetwarzania tekstu (standaryzacja morfologiczna, lematyzacja, embedding).

W ramach tego zadania przeprowadzono testy wybranych bibliotek na zgromadzonym zbiorze danych. Ze względu na cele projektu jako kryteria oceny przyjęto możliwości wybranych bibliotek głównie pod kątem budowania złożonych potoków przetwarzania tekstu z nakierowaniem szczególnie na zadania parsowania zdań, klasyfikacji i rozpoznawania encji. Istotna również była elastyczność bibliotek pod kątem używania różnych implementacji poszczególnych kroków algorytmów, czytelność kodu i popularność oraz wydajność.

Jako benchmark do analizy bibliotek wybrano zadanie klasyfikacji próbki dokumentów ze zbioru treningowego.

Biorąc pod uwagę powyższe kryteria jako bazowa bibliotekę do implementacji kolejnych zadań projektowych wybrano bibliotekę SpaCy. Testy na próbce danych wykazały lepszą wydajność biblioteki SpaCy w podstawowych zadaniach jak parsowanie tekstu i tagowanie części mowy prawdopodobnie ze względu na implementację kluczowych struktur w języku C (cython). Również czytelność kodu w SpaCy jest zdecydowanie lepsza ze względu na stosowanie obiektowego API w porównaniu z NLTK, gdzie wyniki działania są obiektami typu string. Architektura SpaCy pozwala również wykorzystywać w potoku przetwarzania różne rodzaje embeddingu co pozwala wykorzystywać biblioteki word2vec czy FastText jako wymienne komponenty architektury, dlatego na tym etapie nie eliminowano żadnej z wymienionych opcji jako implementacji embeddingu.

Podsumowując wnioski dotyczące wykorzystania poszczególnych bibliotek były następujące:

1. NLTK - odrzucono wykorzystanie tej biblioteki głównie ze względu na wydajność i przejrzystość kodu przy braku przewag w jakości klasyfikacji.
2. SpaCy - przyjęto jako podstawową bibliotekę do wykonywania operacji NLP głównie ze względu na modułowość (możliwość wykorzystywania różnych modeli i algorytmów), obiektowość kodu i wydajność. Jakość klasyfikacji zależy od wybranych modeli, co wykazano w dalszych etapach prac.
3. gensim - biblioteka dedykowana głównie do zadań typu topic modelling ale dająca możliwość wykorzystania własnych embedding-ów również w połączeniu z SpaCy. W dalszych pracach zrezygnowano jednak ze stosowania embeddingu (o czym dalej).
4. word2vec - testowano różne implementacje embeddingu typu word2vec (google, clarin, gensim). Architektura zakładała wykorzystanie istniejącego embeddingu do generowania wektorów wejściowych do warstwy klasyfikacyjnej. Jako warstwę klasyfikacyjną testowano zarówno sieć CNN jak i klasyfikatory regresyjne. Ze względu na jakość danych (prawdopodobnie ze względu na błędy ocr w słowach) nie uzyskano znacząco lepszych wyników klasyfikacji niż dla metod regresyjnych bazujących na modelach Bag Of Words.
5. FastText - testowano różne implementacje embeddingu typu FastTextc (google, clarin, gensim). Architektura zakładała wykorzystanie istniejącego embeddingu do generowania wektorów wejściowych do warstwy klasyfikacyjnej. Jako warstwę klasyfikacyjną testowano zarówno sieć CNN jak i klasyfikatory regresyjne. Jakość klasyfikacji była lepsza niż dla modeli word2vec prawdopodobnie ze względu na to, że algorytm ten dokonuje embeddingu na poziomie znaków a nie całych słów, dzięki czemu w pewnym stopniu eliminowane są błędy literowe wynikające z niedoskonałości procesu OCR. Nadal jednak nie uzyskano znacząco lepszych wyników klasyfikacji niż dla metod regresyjnych bazujących na modelach Bag Of Words.

Ewaluacji poddane zostały narzędzia takie jak: SpaCy, NLTK, gensim, word2vec, fastText oraz nastąpiła parametryzacja wybranych narzędzi dla języka polskiego i uczenie na bazie zgromadzonego korpusu dokumentów.

Założeniem projektowanego rozwiązania było prawidłowe korelowanie dokumentu pojawiającego się na wejściu z innymi dokumentami powiązanymi semantycznie. Aby to było możliwe konieczne było rozróżnienie różnych rodzajów dokumentów i przypisanie do danego dokumentu właściwej klasy, aby w następnej kolejności uruchomić odpowiedni proces weryfikacji. Jest to z punktu widzenia NLP zadanie klasyfikacji i takie zadanie zostało wybrane jako podstawowy problem badawczy na tym etapie. W tym celu ze zgromadzonych w poprzednim etapie dokumentów wydzielono zbiór, który zawierał w sumie 18 różnych klas dokumentów:

Nazwa kategorii	Liczba dokumentów
wniosek zfron	7804
faktura	5147
umowa zlecenie	2221
umowa o pracę tymczasową	1793
karta czasu pracy pracownika	1766
rachunki do umów - zlecenia	1698
L-4	1445
lista obecności	1038
roczna karta czasu pracy	839
rozwiązanie umowy zlecenia	827
umowa	653

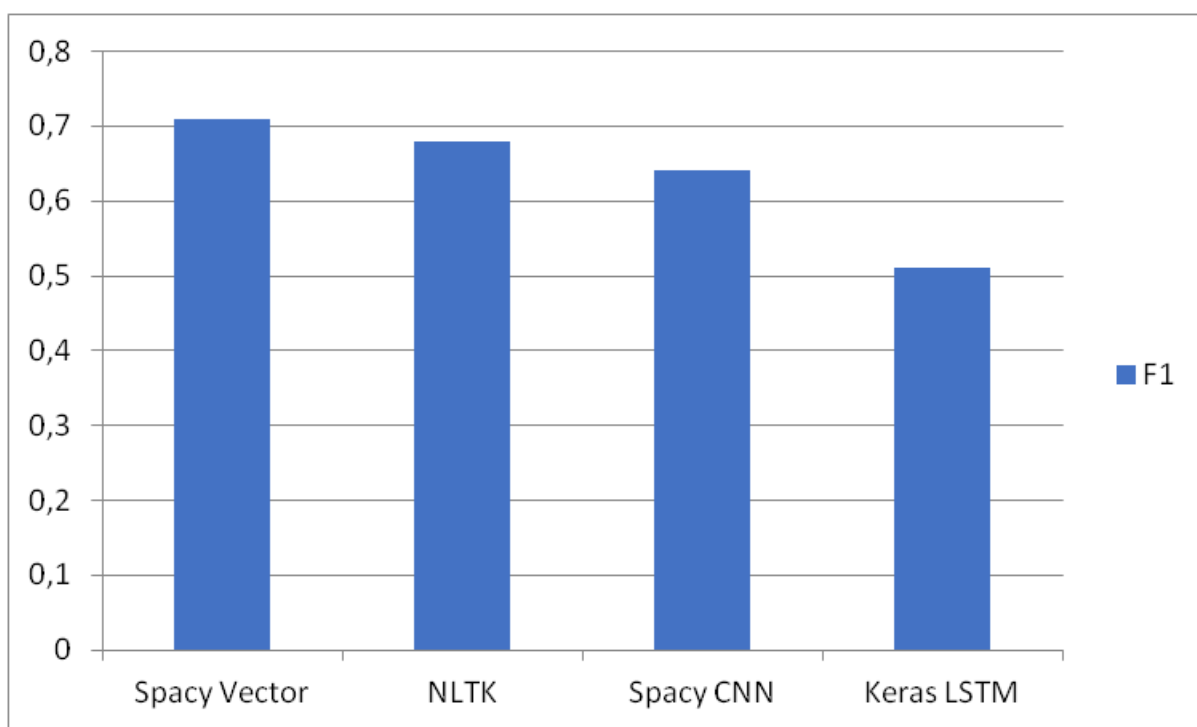
umowa o pracę	517
badania okresowe	301
wniosek urlopowy	161
wniosek o zatrudnienie	137
skierowanie na badania lekarskie	122
zgoda na pracę	99
opis stanowiska	95

Ze względu na kluczowy dla projektu temat weryfikacji faktur z umowami wydzielono podzbiór umów podzielony dodatkowo na podklasy:

Nazwa kategorii	Liczba dokumentów
Umowa dostawy	89
Umowa usługi	3
Umowa depozytowa	3
Umowa dzierżawy	5

Na tak uzyskanym zbiorze przeprowadzono szereg eksperymentów polegających na przeprowadzeniu zadania klasyfikacji dokumentów. Przy założeniu podziału zbioru na testowy i treningowy w proporcjach 20/80 testowano różne konfiguracje parametrów różnych modeli ML w celu wyboru takich konfiguracji modeli, która da najlepsze efekty w postaci statystyki F1 opisanego zadania klasyfikacji.

W trakcie prac wstępnych przeprowadzono ewaluację bibliotek przetwarzania języka naturalnego oraz możliwość wykorzystania różnych modeli na zadaniu klasyfikacji danych. Na próbce danych testowano trening i prognozowanie klasy dokumentu. Przeprowadzone badania nie wykazały istotnych różnic statystycznych w skuteczności działania różnych bibliotek. Należy zauważyć, że takie pakiety NLP jak NLTK i Spacy oferują w dużej części implementację tych samych algorytmów a różnią się głównie wewnętrzną architekturą i wydajnością działania i sposobem programowania. Specyficzne zagadnienie użycia sieci neuronowych typu CNN jest możliwe bezpośrednio z poziomu biblioteki Spacy. Dodatkowo przeprowadzono próby z wykorzystaniem biblioteki Keras i modelu LSTM. Poniższy wykres przedstawia wartość średniej harmonicznej F1 dla klasyfikacji próbki danych.



Wnioski z przeprowadzonych prób są następujące:

- Model Spacy Vector oparty na wektoryzacji BOW i modelu LogReg miał skuteczność prawie identyczną jak taki sam model (Bag Of Words + Logistic Regression) w bibliotece NLTK. Natomiast czas trenowania i inferencji dla biblioteki NLTK dla próbki danych obejmujących losowy podzbiór ze zbioru dokumentów był średnio ponad 4x dłuższy.
- Model SpacyCNN oparty na architekturze sieci neuronowych nie miał żadnych przewag w stosunku do modeli regresyjnych - skuteczność mierzona F1 była nawet niższa, a czas uczenia i inferencji około 2x dłuższy.
- Dla porównania testowano również model oparty na architekturze LSTM z implementacją za pomocą biblioteki Keras, lecz uzyskane tutaj wyniki były znacznie gorsze i zdecydowano o nie wykorzystywaniu tych rozwiązań. Złożone modele neuronowe dla przetwarzanych w projekcie danych wykazywały trudność w uczeniu i szybko osiągały wysycenie przy którym krzywa uczenia nie wykazywała postępów w

kolejnych epokach treningu prawdopodobnie ze względu na dużą złożoność modelu w odniesieniu do ilości danych.

W konsekwencji przeprowadzonych prób jako podstawową bibliotekę do zadań NLP w opracowywanym produkcie można było dalej rozważać bibliotekę SpaCy i NLTK. Przeprowadzone testy wydajności dla typowych zadań NLP zdecydowanie wskazały jednak bibliotekę SpaCy jako najlepszy wybór.

Wyniki dla próbki danych o rozmiarze około 10kB:

Biblioteka	Tokenizer	Tagging	Parsing
spaCy	0.2ms	2ms	15ms
NLTK	4,5ms	582ms	brak wsparcia

Dodatkowo przeprowadzono jeszcze porównanie działania algorytmów NER dostępnych w obu bibliotekach “out of the box” i tutaj również biblioteka SpaCy oferuje lepsze rozwiązania.

Biblioteka	F1
spaCy	0.69
NLTK	0.58

Następnie już przy użyciu biblioteki SpaCy i wybranych plugin’ów do tej biblioteki testowano konkretne konfiguracje modeli. Pod uwagę brano następujące elementy wpływające na strukturę modelu:

1. Klasyfikacja na całym dokumencie lub na jego podsumowaniu
2. Sposób ekstrakcji cech: BOW, HASH, TFIDF, FastText, Word2Vec
3. Model uczenia maszynowego: LogReg, SGD, SVC, BERT, CNN, GradientBoosting, KNN, LogReg, RandomForrest, Transformer.

Istnieje bardzo duża ilość możliwych kombinacji wskazanych wyżej elementów. Dodatkowo każdy z algorytmów posiada wiele parametrów, które mogą mieć duży wpływ na wynik. Nie wszystkie kombinacje wskazanych elementów mają sens i takie zostały wyeliminowane z eksperymentów. W celu porównania i wyboru najlepszych kombinacji zaprojektowano zestaw eksperymentów wykorzystujących wspomniane zadanie klasyfikacji. W tym celu zbudowano dedykowane do eksperymentów środowisko umożliwiające sprawne przeprowadzanie eksperymentów i gromadzenie wyników:

1. Serwer repozytorium plików z dostępem do danych wstępnie przetworzonych w poprzednim zadaniu obejmujących treść dokumentów i powiązane metadane, w tym klasę dokumentu.
2. Serwer uczenia maszynowego zoptymalizowany.
3. Serwer koordynujący przebieg eksperymentu umożliwiający podgląd i wizualizację procesu oraz zbieranie danych wynikowych.
4. Właściwe skrypty trenujące i walidujące testowane modele.

Mieszanie podstawowych elementów takich jak sposób ekstrakcji cech oraz dobór modelu ML przeprowadzono w sposób kombinatoryczny. Natomiast do optymalizacji doboru parametrów poszczególnych modeli wykorzystano gotowy algorytm tuningu hiperparametrów. Aby osiągnąć wykonywalność tak zaprojektowanego eksperymentu konieczne było wykorzystanie serwera z dostępem do GPU.

Przetestowano około 2 tys. kombinacji kluczowych elementów i parametrów modeli.

Wyniki eksperymentów gromadzone były w pliku arkusza kalkulacyjnego:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	F1	dane	ekstraktor ce	klasyfikat	param	klasyfikatora								
2	0.3948	pełny tekst dokumentów	wszystkie dokumenty	BOW	LogReg	{'C': 0.1, 'solver': 'sag', 'penalty': 'l2', 'multi_class': 'ovr', 'tol': 0.01}								
3	0.8206	pełny tekst dokumentów	wszystkie dokumenty	BOW	SVC	{'C': 3.0, 'multi_class': 'ovo'}								
4	0.1846	pełny tekst dokumentów	wszystkie dokumenty	BOW	SGD	{'loss': 'modified_huber', 'penalty': 'l1', 'l1_ratio': 0.007, 'alpha': 0.8, 'tol': 0.06}								
5	0.7634	pełny tekst dokumentów	wszystkie dokumenty	TFIDF	LogReg	{'C': 0.1, 'solver': 'sag', 'penalty': 'l2', 'multi_class': 'ovr', 'tol': 0.01}								
6	0.8532	pełny tekst dokumentów	wszystkie dokumenty	TFIDF	SVC	{'C': 3.0, 'multi_class': 'ovo'}								
7	0.1394	pełny tekst dokumentów	wszystkie dokumenty	TFIDF	SGD	{'loss': 'modified_huber', 'penalty': 'l1', 'l1_ratio': 0.007, 'alpha': 0.8, 'tol': 0.06}								
8	0.7577	pełny tekst dokumentów	wszystkie dokumenty	HASH	LogReg	{'C': 0.1, 'solver': 'sag', 'penalty': 'l2', 'multi_class': 'ovr', 'tol': 0.01}								
9	0.8563	pełny tekst dokumentów	wszystkie dokumenty	HASH	SVC	{'C': 3.0, 'multi_class': 'ovo'}								
10	0.1394	pełny tekst dokumentów	wszystkie dokumenty	HASH	SGD	{'loss': 'modified_huber', 'penalty': 'l1', 'l1_ratio': 0.007, 'alpha': 0.8, 'tol': 0.06}								
11	0.1532	pełny tekst dokumentów	wszystkie dokumenty	Word2Vec	SVC	{'C': 3.0, 'multi_class': 'ovo'}								
12	0.1394	pełny tekst dokumentów	wszystkie dokumenty	FastText	SVC	{'C': 3.0, 'multi_class': 'ovo'}								
13	0.7686	pełny tekst dokumentów	max. 5000 dokumentów	TFIDF	SVC	{'C': 3.0, 'multi_class': 'ovo'}								
14	0.8516	pełny tekst dokumentów	wszystkie dokumenty		CNN	{'max_tokens': 5000, 'output_sequence_length': 512, 'embedding_dim': 64, 'dropout': 0.25, 'kernel_size': 3, 'num_filters': 128, 'filter_sizes': [2, 3, 4]}								
15	0.8099	pełny tekst dokumentów	wszystkie dokumenty		Transformer	{'optimize': 'adam', 'learning_rate': 0.001, 'embedding_dim': 32, 'num_heads': 2, 'dense_dim': 32, 'dropout': 0, 'optimizer': 'adam_weight_decay', 'learning_rate': 2e-05, 'max_sequence_length': 512}								
16	0.8562	pełny tekst dokumentów	wszystkie dokumenty		BERT	{'dropout': 0, 'optimizer': 'adam_weight_decay', 'learning_rate': 2e-05, 'max_sequence_length': 512}								
17	0.7671	1 zdanie podsumowania	wszystkie dokumenty	BOW	SVC	{'C': 3.0, 'multi_class': 'ovo'}								
18	0.1394	1 zdanie podsumowania	wszystkie dokumenty	BOW	SGD	{'loss': 'modified_huber', 'penalty': 'l1', 'l1_ratio': 0.007, 'alpha': 0.8, 'tol': 0.06}								
19	0.6509	1 zdanie podsumowania	wszystkie dokumenty	TFIDF	LogReg	{'C': 0.1, 'solver': 'sag', 'penalty': 'l2', 'multi_class': 'ovr', 'tol': 0.01}								
20	0.7742	1 zdanie podsumowania	wszystkie dokumenty	TFIDF	SVC	{'C': 3.0, 'multi_class': 'ovo'}								
21	0.1394	1 zdanie podsumowania	wszystkie dokumenty	TFIDF	SGD	{'loss': 'modified_huber', 'penalty': 'l1', 'l1_ratio': 0.007, 'alpha': 0.8, 'tol': 0.06}								
22	0.6473	1 zdanie podsumowania	wszystkie dokumenty	HASH	LogReg	{'C': 0.1, 'solver': 'sag', 'penalty': 'l2', 'multi_class': 'ovr', 'tol': 0.01}								
23	0.7738	1 zdanie podsumowania	wszystkie dokumenty	HASH	SVC	{'C': 3.0, 'multi_class': 'ovo'}								
24	0.1394	1 zdanie podsumowania	wszystkie dokumenty	HASH	SGD	{'loss': 'modified_huber', 'penalty': 'l1', 'l1_ratio': 0.007, 'alpha': 0.8, 'tol': 0.06}								

Podstawowa interpretacja przeprowadzonych eksperymentów sprowadza się do posortowania uzyskanych wyników względem wartości funkcji F1 świadczącej o jakości klasyfikacji:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	F1	dane	ekstraktor	klasyfikator	parametry	klasyfikatora							
2	0.8569682	pełny tekst dokumentów	wszystkie dokumenty	TFIDF	KNN	{'n_neighbors': 3, 'weights': 'distance', 'algorithm': 'ball_tree', 'leaf_size': 10}							
3	0.8569682	pełny tekst dokumentów	wszystkie dokumenty	TFIDF	KNN	{'n_neighbors': 3, 'weights': 'distance', 'algorithm': 'ball_tree', 'leaf_size': 20}							
4	0.8569682	pełny tekst dokumentów	wszystkie dokumenty	TFIDF	KNN	{'n_neighbors': 3, 'weights': 'distance', 'algorithm': 'ball_tree', 'leaf_size': 30}							
5	0.8569682	pełny tekst dokumentów	wszystkie dokumenty	TFIDF	KNN	{'n_neighbors': 3, 'weights': 'distance', 'algorithm': 'ball_tree', 'leaf_size': 40}							
6	0.8569682	pełny tekst dokumentów	wszystkie dokumenty	TFIDF	KNN	{'n_neighbors': 3, 'weights': 'distance', 'algorithm': 'ball_tree', 'leaf_size': 50}							
7	0.8569682	pełny tekst dokumentów	wszystkie dokumenty	TFIDF	KNN	{'n_neighbors': 3, 'weights': 'distance', 'algorithm': 'kd_tree', 'leaf_size': 10}							
8	0.8569682	pełny tekst dokumentów	wszystkie dokumenty	TFIDF	KNN	{'n_neighbors': 3, 'weights': 'distance', 'algorithm': 'kd_tree', 'leaf_size': 20}							
9	0.8569682	pełny tekst dokumentów	wszystkie dokumenty	TFIDF	KNN	{'n_neighbors': 3, 'weights': 'distance', 'algorithm': 'kd_tree', 'leaf_size': 30}							
10	0.8569682	pełny tekst dokumentów	wszystkie dokumenty	TFIDF	KNN	{'n_neighbors': 3, 'weights': 'distance', 'algorithm': 'kd_tree', 'leaf_size': 40}							
11	0.8569682	pełny tekst dokumentów	wszystkie dokumenty	TFIDF	KNN	{'n_neighbors': 3, 'weights': 'distance', 'algorithm': 'kd_tree', 'leaf_size': 50}							
12	0.8569682	pełny tekst dokumentów	wszystkie dokumenty	TFIDF	KNN	{'n_neighbors': 3, 'weights': 'distance', 'algorithm': 'brute', 'leaf_size': 10}							
13	0.8569682	pełny tekst dokumentów	wszystkie dokumenty	TFIDF	KNN	{'n_neighbors': 3, 'weights': 'distance', 'algorithm': 'brute', 'leaf_size': 20}							
14	0.8569682	pełny tekst dokumentów	wszystkie dokumenty	TFIDF	KNN	{'n_neighbors': 3, 'weights': 'distance', 'algorithm': 'brute', 'leaf_size': 30}							
15	0.8569682	pełny tekst dokumentów	wszystkie dokumenty	TFIDF	KNN	{'n_neighbors': 3, 'weights': 'distance', 'algorithm': 'brute', 'leaf_size': 40}							
16	0.8569682	pełny tekst dokumentów	wszystkie dokumenty	TFIDF	KNN	{'n_neighbors': 3, 'weights': 'distance', 'algorithm': 'brute', 'leaf_size': 50}							
17	0.8563	pełny tekst dokumentów	wszystkie dokumenty	HASH	SVC	{'C': 3.0, 'multi_class': 'ovo'}							
18	0.8562	pełny tekst dokumentów	wszystkie dokumenty		BERT	{'dropout': 0, 'optimizer': 'adam_weight_decay', 'learning_rate': 2e-05, 'max_sequ							
19	0.8532	pełny tekst dokumentów	wszystkie dokumenty	TFIDF	SVC	{'C': 3.0, 'multi_class': 'ovo'}							
20	0.8516	pełny tekst dokumentów	wszystkie dokumenty		CNN	{'max_tokens': 5000, 'output_sequence_length': 512, 'embedding_dim': 64, 'drop							
21	0.8410757	pełny tekst dokumentów	wszystkie dokumenty	TFIDF	KNN	{'n_neighbors': 4, 'weights': 'distance', 'algorithm': 'ball_tree', 'leaf_size': 10}							
22	0.8410757	pełny tekst dokumentów	wszystkie dokumenty	TFIDF	KNN	{'n_neighbors': 4, 'weights': 'distance', 'algorithm': 'ball_tree', 'leaf_size': 20}							
23	0.8410757	pełny tekst dokumentów	wszystkie dokumenty	TFIDF	KNN	{'n_neighbors': 4, 'weights': 'distance', 'algorithm': 'ball_tree', 'leaf_size': 30}							

Z przedstawionych wyników można przedstawić następujące wnioski:

1. Stosowanie algorytmu podsumowywania dokumentów przyspiesza działanie algorytmów (w szczególności proces trenowania klasyfikatorów), lecz ma negatywny wpływ na końcową jakość klasyfikacji. Aby jednak możliwa była praca na pełnym tekście dokumentów konieczny jest dostęp do GPU. W przeciwnym przypadku trenowanie klasyfikatorów zajmuje w praktyce zbyt wiele czasu.
2. Modele uczenia maszynowego bazujące metodach wektoryzacji biorących pod uwagę tylko zawartość zbioru treningowego (np. TFIDF) dają lepsze wyniki i są wydajniejsze obliczeniowo, niż modele bazujące na złożonych modelach językowych, które są pre trenowane na tekstach “natywnie cyfrowych” (czyli źródłach takich jak wikipedia czy inne teksty dostępne cyfrowo w internecie). W szczególności widoczne jest to dla modeli (CNN, BERT), których działanie jest zdecydowanie bardziej czasochłonne, nie dając przy tym znacząco lepszej jakości predykcji. Taka sytuacja wynika bezpośrednio z rodzaju przetwarzanych danych. Teksty, które są przetwarzane przez algorytmy pochodzą z procesu OCR, który to proces wprowadza losowy szum informacyjny w postaci błędów w treści, które z kolei skutkują powstawaniem słów (tokenów), które nie występowały w zbiorach treningowych wykorzystywanych do trenowania złożonych modeli językowych. Przez to zjawisko wyznaczanie wektorów reprezentujących te słowa w procesie tzw. embeddingu jest zaburzone i w praktyce słowa, które występują w dokumencie i zawierają błędy otrzymują reprezentację wektorową odległą od oczekiwanej rzeczywistej reprezentacji tego słowa. To zjawisko jest potwierdzone przez inne badania, np. “An Assessment of the Impact of OCR Noise on Language Models”, Konstantin Todorov and Giovanni Colavizza, i w praktyce **powoduje, że złożone modele językowe mają ograniczone zastosowanie do tego typu danych, w tym również w niniejszym projekcie.**

UWAGA OGÓLNA: Przedstawione w powyższej tabeli wyniki i tak pokazują dość wysoką pozycję algorytmów pre trenowanych wykorzystujących złożone modele językowe oparte na złożonych sieciach neuronowych. W pierwszych etapach realizacji projektu wyniki dla tych

algorytmów były znacznie gorsze (wartość F1 niższa o 0,15-0,25). Tabela zawiera wyniki z samego końca projektu, które są lepsze niż początkowe dlatego, że proces usprawniania całego rozwiązania, w tym procesu wyznaczania reprezentacji tekstowej dokumentów (OCR, normalizacja tekstu) trwał nieprzerwanie przez cały projekt w modelu iteracyjnym. **Pokazuje to bardzo wyraźnie, że nie jest możliwe prowadzenie tego typu projektów w sposób kaskadowy, gdzie zamykając jedno zadanie opisuje się jego wyniki i przechodzi się do następnego. Takie podejście jest pozbawione sprzężenia zwrotnego, które uniemożliwia ciągle doskonalenie całości rozwiązania i wyciąganie wniosków z kolejnych działań.** Podejście iteracyjne jest powszechnie uznanym, typowym podejściem do innowacyjnych projektów w branży IT. Należy jednak pamiętać, że skutkuje to de facto rozmyciem granic pomiędzy poszczególnymi “zadaniami”, gdyż te “zadania” są traktowane raczej jako cele do osiągnięcia i są realizowane niejako równolegle, a nie w sposób kaskadowy na zasadzie przechodzenia od jednego zadania do kolejnego zgodnie z ustalonym z góry harmonogramem.

4. Ewaluacja narzędzi, badania i wybór metody klasyfikacji semantycznej zdań.

Zgodnie z założeniami projektu na tym etapie dokonano ewaluacji możliwości wykorzystania złożonych sieci neuronowych do realizacji zadań przewidzianych w projekcie. Na dzień realizacji tego etapu prac za najbardziej zaawansowane narzędzia NLP tego typu uważane były tzw. transformery, których przykładem jest BERT.

1. BERT - do testów wykorzystano gotowe, wytrenowane transformery typu BERT. Po wstępnej selekcji wybrano do testów implementacje Slavic-BERT i HerBERT. Aby zrealizować zadanie klasyfikacji modele wykorzystano do generowania embeddingu a następnie zastosowano klasyfikator regresyjny. Żaden z modeli nie wykazał się lepszą skutecznością niż inne metody reprezentacji cech (Bag Of Words etc.) ze względów wcześniej wskazanych (błędy w słowach w wyniku działania OCR). Dodatkowo problemem była duża objętość modeli i długi czas inferencji. Z tych względów zdecydowano nie wykorzystywać modeli transformerów w projekcie.
2. CNN/keras - W trakcie ewaluacji samej biblioteki SpaCy uzyskano wyniki wskazujące, że modele wykorzystujące do klasyfikacji głębokie sieci neuronowe nie mają przewagi nad modelami regresyjnymi. Prawdopodobnie wpływ na to mają dwa czynniki: ilość i jakość danych. Jakość rzeczywistych danych, które były wykorzystywane w projekcie, zdecydowanie odbiega od danych wykorzystywanych w większości publicznie dostępnych testów, które bazują na danych, które są “natywnie cyfrowe” - np. teksty pozyskiwane z Wikipedii. Okazuje się, że dla danych rzeczywistych pochodzących z procesu OCR proste modele regresyjne dają wyniki podobne a nawet lepsze od modeli bardziej złożonych a jednocześnie są szybsze w trenowaniu, mają mniejszy rozmiar i szybciej zwracają wynik, co lepiej predysponuje te modele do wykorzystania w procesach biznesowych.
3. Clarin-pl - modele wytworzone w projekcie Clarin-PL cechują się podobnymi problemami jak wskazane wcześniej dla wszystkich transformerów. Jednak bliższa analiza wykazała, że dobre wyniki daje specjalizowany model NER. W związku z tym

podjęto decyzje o zintegrowaniu w projekcie modelu PolDeepNER2 w celu delegowania zadania ekstrakcji encji. Przykładowy wynik ekstrakcji:

Fly On The Cloud Sp. z o.o. z siedzibą we Wrocławiu przy ul. Rzeźniczej 32/33, 50-130 Wrocław, wpisana do rejestru przedsiębiorców prowadzonego przez Sąd Rejonowy dla Wrocławia Fabrycznej, Wydział VI Krajowego Rejestru Sądowego, pod numerem KRS 0000500884, wysokość kapitału zakładowego 10.000,00 zł, NIP 8971797086, REGON: 022370270 reprezentowana przez: Piotra Wieczko - Prezesa Zarządu (dalej: FOTC) a Extreme Robotics SPÓŁKA Z OGRANICZONĄ ODPOWIEDZIALNOŚCIĄ NIP 9671434894 z siedzibą w UL. BAGNO 2 / 73, 00-112 Warszawa, Polska, PL reprezentowaną przez: Michał Wolf (dalej: Klient)

W ramach zadania prowadzono ewaluację wybranych bibliotek (CNN/keras, BERT, clarin-pl) oraz wybrane zostały algorytmy do dalszych testów.

Celem klasyfikacji semantycznej zdań było wyłuskanie z treści dokumentów tych zdań, które zawierają dane i reguły, które będą wykorzystywane do walidacji krzyżowej dokumentów. Z analizy tego zagadnienia dziedzinowego w kontekście rodzajów dokumentów, na które ukierunkowany był projekt, jasno wynika, że walidacja krzyżowa musi uwzględniać operacje porównywania atrybutów metadanych dokumentów. Pierwszym problemem do rozwiązania było w związku z tym identyfikowanie zdań, które zawierają te metadane i powiązane z nimi reguły.

W zgromadzonych zbiorach danych udało się określić wartości metadanych dla kluczowych atrybutów dokumentów. Ponieważ dokumentem, do którego odnosi się walidacja jest umowa, to jako bazowy zbiór reguł wybrano atrybuty tego dokumentu:

Metadana	Opis
startDate	data rozpoczęcia obowiązywania umowy
endDate	data zakończenia obowiązywania umowy
noticePeriod	okres wypowiedzenia umowy
signDate	data podpisania umowy
nip	numer NIP drugiej strony umowy
proceedingsNumber	numer postępowania
netValue	wartość netto umowy
grossValue	wartość brutto umowy
accountNumber	numer konta
number	numer umowy
agreementObject	przedmiot umowy
payment	termin opłat
installmentPayment	płatność ratalna (tak/nie)
frequencyOfPayment	częstotliwość opłat

W oryginalnych zbiorach danych dostępna jest treść dokumentów i wartości metadanych. Aby możliwe było traktowanie zadania jako zadania klasyfikacji konieczne było przetworzenie zbioru danych w taki sposób, że z oryginalnej treści dokumentów zostały zidentyfikowane miejsca wystąpień metadanych i zostało wyłuskane otoczenie tekstowe dla tej metadanej. Założenie było takie, że możliwe jest wytrenowanie klasyfikatora, który będzie trenowany do prognozowania rodzaju metadanych występujących w tekście. Na tym etapie natrafiono na szereg problemów związanych z jakością danych:

1. Niezgodność metadanych z wartościami występującymi w tekście ze względu na format. Daty, kwoty czy numery wpisane w treści umowy nie są wpisywane w ujednoliconych formatach. Z kolei wartości wpisywane w metadanych są standaryzowane. Jednak z tego wynika problem taki, że mając wartość metadanej

próba lokalizacji tej wartości w tekście często kończy się jako False Negative - wartości nie znaleziono pomimo tego, że występuje w tekście.

2. Niezgodność metadanych z wartościami występującymi w tekście ze względu na błędy OCR. Ze względu na błędy OCR również pojawia się prawdopodobieństwo wystąpienia błędu typu False Negative, jednak charakter tych błędów jest inny: nawet jeśli format danych jest zgodny to w treści dokumentu może wystąpić błąd literowy, który będzie powodował niezgodność danych.

Aby rozwiązać ten problem konieczne było podjęcie szeregu działań zaradczych:

1. Zastosowanie podejścia kombinatorycznego dla wartości dat i numerycznych polegające na wyszukiwaniu w tekście wartości w jednym z kilku najczęściej występujących formatów.
2. Zastosowanie podejścia heurystycznego polegającego na wyluskiwaniu potencjalnych wartości w treści dokumentów na bazie wzorców tekstowych i zastosowanie rozmytego porównywania na bazie odległości Levenshteina, aby wybrać z kandydatów te tokeny, które są najbardziej "podobne" do poszukiwanych metadanych.
3. Zastosowanie podejścia słownikowego polegającego na walidacji znalezionych wartości tokenów w z góry określonych słownikach (dla typów danych, dla których jest to możliwe, np NIP, numer umowy, kod pocztowy itd.).
4. Zastosowanie weryfikacji ręcznej próbki danych w celu uzyskania "złotego standardu" poprawnych metadanych przynajmniej dla próbki danych.

Po wykonaniu tych prac udało się uzyskać zbiór treningowy, który zawierał fragmenty tekstu wraz z przypisaną odpowiednią klasą. Na tym etapie było możliwe potraktowanie zadania rzeczywiście jako zadania klasyfikacji. Ogólna metodyka, jaka została zastosowana była podobna do stosowanej wcześniej. Zostało przygotowanie środowiska do przeprowadzenia cyklu eksperymentów mających na celu sprawdzanie różnych kombinacji sposobu ekstrakcji cech (wektoryzacji), modelu predykcyjnego oraz hiper-parametrów tego modelu. **Stosując opracowaną wcześniej metodykę przeprowadzono prawie 2 tys. eksperymentów.** Jako metodę oceny jakości modelu przyjęto wartość F1 klasyfikatora.

Uzyskane wyniki przedstawione są w zamieszczonej poniżej tabeli:

	A	B	C	D	E
1	Rodzaj encji	F1	Ekstraktor cech	Klasyfikator	Parametry klasyfikatora
2	endDate	0.8	BOW	LogReg	{'C': 0.001, 'solver': 'lbfgs', 'penalty': 'l2'}
3	endDate	0.8	BOW	SupportVector	{'C': 1.0, 'kernel': 'rbf', 'decision_function_shape': 'ovo'}
4	endDate	0.8	HASH	LogReg	{'C': 0.001, 'solver': 'lbfgs', 'penalty': 'l2'}
5	endDate	0.8	TFIDF	LogReg	{'C': 0.001, 'solver': 'lbfgs', 'penalty': 'l2'}
6	endDate	0.8	TFIDF	SupportVector	{'C': 1.0, 'kernel': 'rbf', 'decision_function_shape': 'ovo'}
7	endDate	0.6982	HASH	SupportVector	{'C': 1.0, 'kernel': 'rbf', 'decision_function_shape': 'ovo'}
8	grossValue	0.698	BOW	LogReg	{'C': 0.001, 'solver': 'lbfgs', 'penalty': 'l2'}
9	grossValue	0.698	BOW	SupportVector	{'C': 1.0, 'kernel': 'rbf', 'decision_function_shape': 'ovo'}
10	grossValue	0.698	HASH	SupportVector	{'C': 1.0, 'kernel': 'rbf', 'decision_function_shape': 'ovo'}
11	grossValue	0.698	TFIDF	SupportVector	{'C': 1.0, 'kernel': 'rbf', 'decision_function_shape': 'ovo'}
12	netValue	0.6954	BOW	LogReg	{'C': 0.001, 'solver': 'lbfgs', 'penalty': 'l2'}
13	netValue	0.6954	HASH	SupportVector	{'C': 1.0, 'kernel': 'rbf', 'decision_function_shape': 'ovo'}
14	netValue	0.6954	TFIDF	SupportVector	{'C': 1.0, 'kernel': 'rbf', 'decision_function_shape': 'ovo'}
15	grossValue	0.6954	HASH	LogReg	{'C': 0.001, 'solver': 'lbfgs', 'penalty': 'l2'}
16	grossValue	0.6954	HASH	SGD	{'loss': 'log_loss', 'penalty': 'l1', 'alpha': 0.0001}
17	grossValue	0.6954	TFIDF	LogReg	{'C': 0.001, 'solver': 'lbfgs', 'penalty': 'l2'}
18	grossValue	0.6954	TFIDF	SGD	{'loss': 'log_loss', 'penalty': 'l1', 'alpha': 0.0001}
19	netValue	0.6928	BOW	SGD	{'loss': 'log_loss', 'penalty': 'l1', 'alpha': 0.0001}
20	netValue	0.6928	HASH	LogReg	{'C': 0.001, 'solver': 'lbfgs', 'penalty': 'l2'}
21	netValue	0.6928	TFIDF	LogReg	{'C': 0.001, 'solver': 'lbfgs', 'penalty': 'l2'}
22	netValue	0.6928	TFIDF	SGD	{'loss': 'log_loss', 'penalty': 'l1', 'alpha': 0.0001}
23	proceedingsNumber	0.6773	BOW	LogReg	{'C': 0.001, 'solver': 'lbfgs', 'penalty': 'l2'}
24	proceedingsNumber	0.6773	BOW	SupportVector	{'C': 1.0, 'kernel': 'rbf', 'decision_function_shape': 'ovo'}

W trakcie eksperymentów testowano zarówno modele bazujące na głębokich sieciach neuronowych i transformerach (CNN, BERT) jak i inne, prostsze modele.

Podobnie jak w poprzednich przypadkach potwierdziło się, że modele wykorzystujące złożone sieci neuronowe dają gorsze wyniki, niż modele prostsze.

Tak jak wcześniej, powodem tego są zarówno błędy OCR, które powodują powstawanie w danych tokenów (słów), które nie występują w embeddingu oraz prawdopodobnie z różnicy w strukturze tekstów pomiędzy stosowanymi w projekcie danymi biznesowymi a tymi, na których są typowo trenowane złożone modele neuronowe (natywnie cyfrowe teksty internetowe). Jednak za pomocą prostych modeli ML udało się osiągnąć wyniki spełniające założenia kamieni milowych projektu.

Poniżej przedstawiono w tabeli ostateczne (najlepsze) wyniki dla analizowanych rodzajów danych:

Rodzaj encji	F1	Ekstraktor cech	Klasyfikator	Parametry klasyfikatora
endDate	0.821	BOW	LogReg	{'C': 0.001, 'solver': 'lbfgs', 'penalty': 'l2'}
startDate	0.786	BOW	SGD	{'loss': 'log_loss', 'penalty': 'l1', 'alpha': 0.0001}
signDate	0.683	TFIDF	SGD	{'loss': 'log_loss', 'penalty': 'l1', 'alpha': 0.0001}
netValue	0.813	BOW	LogReg	{'C': 0.001, 'solver': 'lbfgs', 'penalty': 'l2'}
grossValue	0.698	BOW	LogReg	{'C': 0.001, 'solver': 'lbfgs', 'penalty': 'l2'}
nip	0.898	BOW	SGD	{'loss': 'log_loss', 'penalty': 'l1', 'alpha': 0.0007}
accountNumber	0.819	HASH	LogReg	{'C': 0.001, 'solver': 'lbfgs', 'penalty': 'l2'}
proceedingsNumber	0.873	BOW	LogReg	{'C': 0.001, 'solver': 'lbfgs', 'penalty': 'l2'}
agreementObject	0.143	TFIDF	SGD	{'loss': 'modified_huber', 'penalty': 'l1', 'alpha': 0.0001, 'l1_ratio': 0.1, 'tol': 0.001}
payment	0.671	HASH	KNN	{'n_neighbors': 4, 'weights': 'uniform', 'algorithm': 'brute', 'leaf_size': 50}
installmentPayment	0.438	TFIDF	RandomForest	{'n_estimators': 150, 'criterion': 'gini', 'min_samples_split': 2, 'min_samples_leaf': 1}

5. Przeprowadzenie właściwych badań mających na celu ustalenie właściwego potoku przetwarzania danych, wybór struktury modelu i parametryzacja algorytmów.

W ramach zadania wykonane zostały testy opracowanych algorytmów na zgromadzonym korpusie dokumentów, ocena wyników, wprowadzono zmiany w parametrach i strukturze modeli oraz powtarzano iteracyjnie doskonalenie modelu.

Zgodnie z przyjętym w projekcie i przedstawionym wcześniej podejściem opisane w zadaniu prace, polegające na testowaniu różnych modeli i tuningowaniu ich parametrów prowadzone były iteracyjnie przez cały okres trwania projektu.

W sumie przeprowadzono kilka tysięcy eksperymentów polegających na przygotowaniu danych, trenowaniu modeli oraz obliczeniu i ocenie metryk dla różnych kombinacji elementów struktury modeli. Uzyskane najlepsze modele zostały uwzględnione jako komponenty ostatecznej architektury rozwiązania.

Na tym etapie przeprowadzono właściwe prace zmierzające do osiągnięcia założonego kamienia milowego. Na bazie przeprowadzonych dotąd badań zdecydowano, że ekstrakcja danych z dokumentów (na przykładzie umów) będzie się odbywać przez zastosowanie autorskiego podejścia będącego rozwinięciem koncepcji NER. Podejście to zakłada następującą konstrukcję algorytmu:

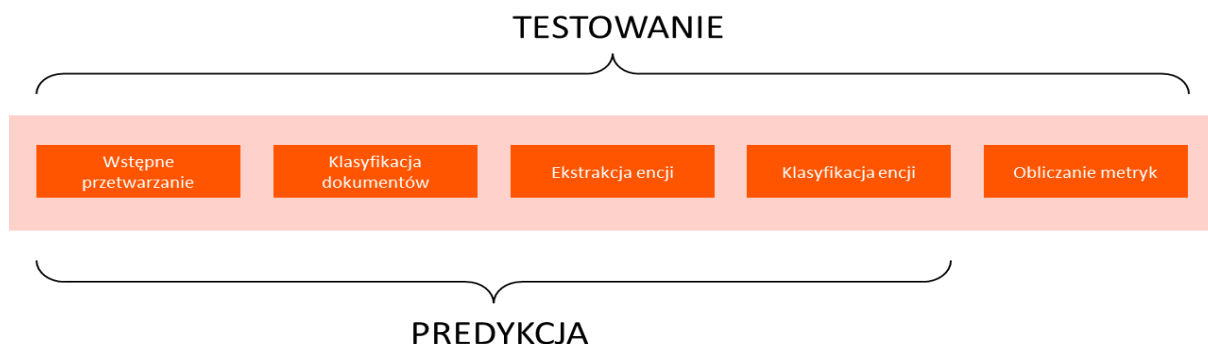
1. Dokumenty wejściowe są wstępnie przetwarzane zgodnie z opracowaną na wcześniejszych etapach procedurą w celu uzyskania cyfrowej reprezentacji zawartości (tekstu) dokumentu.
2. W drugim kroku tekst jest normalizowany w celu uzyskania standardowej formy słów oraz (o ile to możliwe) korekcji błędów OCR).
3. Tak przetworzony dokument jest poddawany klasyfikacji w celu stwierdzenia, czy istnieje dedykowany dla danego rodzaju dokumentu proces ekstrakcji danych. Jeśli tak uruchamiany jest odpowiedni proces.

4. W procesie ekstrakcji danych tekście uruchamiany jest szereg modeli ekstrakcji danych. Na tym etapie uzyskujemy informację o wystąpieniu w tekście bazowego typu danych (data, liczba, osoba itd.) ale jeszcze bez przypisania do właściwego pola metadanych (data sprzedaży czy data płatności). Modele te są dwóch typów:
 - a. Modele reguły (heurystyczne) - modele bazujące na regułach dopasowywania tekstu do wyrażeń regularnych lub Matchera wyrażeń modułu SpaCy. Modele te zostały wytworzone w wyniku analizy danych i są z góry określone dla odpowiednich formatów danych.
 - b. Modele maszynowe wykorzystujące algorytmy NER biblioteki PolDeepNER oraz SpaCy NER.
5. Następnie odczytane wartości danych są przypisywane do pól metadanych związanych z danym typem dokumentu. To zadanie realizują klasyfikatory maszynowe wytworzone w wyniku trenowania na danych treningowych algorytmów klasyfikujących fragmenty tekstu pod kątem występowania w danym fragmencie tekstu szukanej wartości metadanych. Wynikiem działania klasyfikatorów jest przypisanie wartości danych od odpowiedniego "slotu" w obiekcie metadanych.
6. Uzyskane z wszystkich modeli dane są zbierane do postaci dokumentu JSON opisującego metadane dokumentu odczytane z tekstu.

Poniżej przedstawiono wyniki ekstrakcji danych spełniające założenia kamienia milowego dla typu dokumentu "umowa":

Rodzaj encji	F1	Ekstraktor cech	Klasyfikator	Parametry klasyfikatora
endDate	0.821	BOW	LogReg	{'C': 0.001, 'solver': 'lbfgs', 'penalty': 'l2'}
startDate	0.786	BOW	SGD	{'loss': 'log_loss', 'penalty': 'l1', 'alpha': 0.0001}
signDate	0.683	BOW	SGD	{'loss': 'log_loss', 'penalty': 'l1', 'alpha': 0.0001}
zipCode	0.742	BOW	SGD	{'loss': 'log_loss', 'penalty': 'l1', 'alpha': 0.00030000000000000003}
netValue	0.813	BOW	LogReg	{'C': 0.001, 'solver': 'lbfgs', 'penalty': 'l2'}
grossValue	0.698	BOW	LogReg	{'C': 0.001, 'solver': 'lbfgs', 'penalty': 'l2'}
nip	0.898	BOW	SGD	{'loss': 'log_loss', 'penalty': 'l1', 'alpha': 0.0007}
number	0.619	HASH	LogReg	{'C': 0.001, 'solver': 'lbfgs', 'penalty': 'l2'}
proceedingsNumber	0.873	BOW	LogReg	{'C': 0.001, 'solver': 'lbfgs', 'penalty': 'l2'}

W dalszej części dokumentu opisano bardziej szczegółowo przeprowadzone prace i uzyskane wyniki.



Analiza danych

1. Zbiór danych Korespondencja

Zbiór danych składa się z 26 663 dokumentów, należących do 18 kategorii:

Nazwa kategorii	Liczba dokumentów
wniosek zfron	7804
faktura	5147
umowa zlecenie	2221
umowa o pracę tymczasową	1793
karta czasu pracy pracownika	1766
rachunki do umów - zlecenia	1698
L-4	1445
lista obecności	1038

roczna karta czasu pracy	839
rozwiązanie umowy zlecenia	827
umowa	653
umowa o pracę	517
badania okresowe	301
wniosek urlopowy	161
wniosek o zatrudnienie	137
skierowanie na badania lekarskie	122
zgoda na pracę	99
opis stanowiska	95

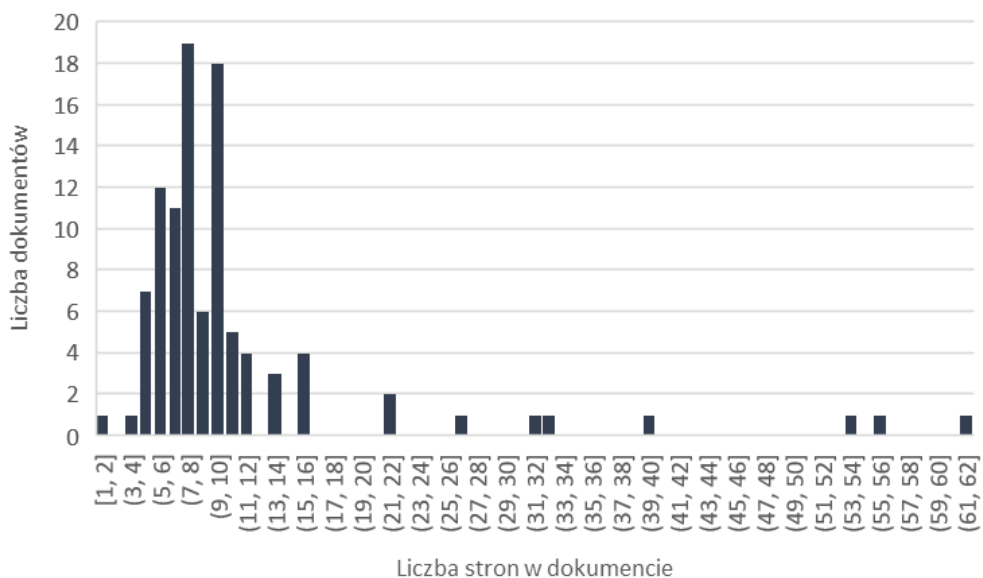
W zbiorze występuje 28 rodzajów metadanych:

Metadana	Opis	Liczba wystąpień w zbiorze danych
number	Numer dokumentu	26662
postCompany	Nazwa firmy kurierskiej	3427
sender_name	nazwa nadawcy	26663

sender_nip	Numer NIP nadawcy	26663
sender_phone	Numer telefonu nadawcy	26663
sender_address_city	Miasto nadawcy	26663
sender_address_street	Ulica nadawcy	26663
sender_address_house	Numer budynku nadawcy	26663
sender_address_flat	Numer lokalu/mieszkania nadawcy	26663
sender_address_zipCode	Kod pocztowy nadawcy	26663
sender_address_province	Województwo nadawcy	26663
address_city	Miasto odbiorcy	26663
address_street	Ulica odbiorcy	26663
address_house	Numer budynku odbiorcy	26663
address_flat	Numer lokalu/mieszkania odbiorcy	26663
address_zipCode	Kod pocztowy odbiorcy	26663
address_province	województwo odbiorcy	26663
companies	Strony umowy	26754

2. Zbiór danych Umowy przetargowe 100

Zbiór danych składa się ze 100 dokumentów – umów przetargowych. Dokumenty różnią się liczbą stron – poniżej przedstawiony rozkład długości dokumentów.



W zbiorze jest 5 kategorii dokumentów:

Nazwa kategorii	Liczba dokumentów
Umowa dostawy	89
Umowa usługi	3
Umowa depozytowa	3
Umowa dzierżawy	5

W zbiorze występuje 28 typów metadanych:

Metadana	Opis	Liczba wystąpień w zbiorze danych
startDate	data rozpoczęcia obowiązywania umowy	100
endDate	data zakończenia obowiązywania umowy	98
noticePeriod	okres wypowiedzenia umowy	0
signDate	data podpisania umowy	100
nip	numer NIP drugiej strony umowy	95
proceedingsNumber	numer postępowania	99
netValue	wartość netto umowy	100
grossValue	wartość brutto umowy	100
name	nazwa drugiej strony umowy	100
phone	numer telefonu drugiej strony umowy	0
zipCode	kod pocztowy drugiej strony umowy	98
province	województwo drugiej strony umowy	97
city	miasto drugiej strony umowy	98
street	ulica drugiej strony umowy	98
house	numer budynku drugiej strony umowy	25

flat	numer lokalu/mieszkania drugiej strony umowy	1
number	numer umowy	100
agreementObject	przedmiot umowy	100
payment	termin opłat	100
installmentPayment	płatność ratałna (tak/nie)	100
frequencyOfPayment	częstotliwość opłat	3

2. Wstępne przetwarzanie danych

Oba zbiory danych zostały podzielone na zbiory treningowe (80% dokumentów) i zbiory testowe (20% dokumentów).

1. Ekstrakcja streszczenia/podsumowania

1. Filtracja tokenów

Przy użyciu biblioteki SpaCy i pretrenowanego modelu „pl_core_news_lg” treść dokumentów została zanotowana. W ten sposób tekst podzielony został na słowa/znaki (tokeny) – każde z odpowiadającymi mu anotacjami/tagami.

Z biblioteki SpaCy pobrana została lista tokenów, które uznawane są za nieistotne przy rozwiązywaniu problemów przetwarzania języka naturalnego, takich jak klasyfikacja dokumentów. Lista ta zawiera 381 znormalizowanych słów, tak zwanych stopwords, przykładowo:

"cała", "dla", "przy", "sposob", "wielu", "o", "ci", "niemu", "ktora", "jakichś", "tel", "kto", "xv", "nasz", "niego", "każdy", "powinni", "prawie", "ja", "godz", "czyli", "to", "mną", "dosc", "bedzie", "zaś", "xii", "dwaj", "jeden", "moje", "totez", "ona", "której", "aby", "gdyz", "rowniez", "żadne", "bynajmniej", "mna", "oni", "tylko", "mamy", "kierunku"

Dla każdego dokumentu tokeny, które:

- nie znajdują się na liście stopwords
- nie są znakami interpunkcji

- zostały otagowane jako jedna z części mowy: PROPN, ADJ, NOUN, VERB

zostały wybrane jako słowa kluczowe. Wielkie litery w słowach kluczowych zamieniane są na małe litery.

2. Filtracja zdań

Dla każdego słowa kluczowego obliczona została liczba jego wystąpień w dokumencie. Po znormalizowaniu są to wagi określające znaczenie danego słowa w dokumencie.

Dla każdego zdania w dokumencie obliczona została waga stanowiąca sumę wag słów kluczowych w owym zdaniu. Zdanie charakteryzujące się największą wagą zostało wybrane jako streszczenie/podsumowanie dokumentu.

3. Normalizacja

Ze streszczeń/podsumowań usunięte zostały znaki interpunkcji oraz wielokrotne spacje.

Normalizacja tokenów w podsumowaniu:

- Tokeny, które są rozpoznawane przez model SpaCy NER jako encje typu persName, date lub orgName, zamienione zostały na symbole zastępcze odpowiadające ich typom.
- Symbole zastępcze występujące jeden po drugim połączone zostały w jeden.
- Tokeny, które są na liście stopwords, zostały usunięte.
- Pozostałe tokeny zastąpione zostały przez ich znormalizowaną formę (lemma).

Dla każdego dokumentu lista znormalizowanych tokenów połączona została w jeden obiekt klasy string, tworząc w ten sposób znormalizowane podsumowanie.

PRZYKŁAD:

Treść dokumentu po użyciu OCR:

OMEGA Faktura VAT nr 2/06/16 strona 1/1 Faktura VAT nr2/06/16
Oryginał Miejs : BKUROWIEEFA SYSTEM | » Data wystawienia : AM!
2016 SPRZEDAWCA : : :* NABYWCA . ——— OMEGA SYSTEM
KRZYSZTOF OMELKO CLARSYSTEM Ss. A. PASIKUROWICE UL.
LEŚNA 7 UL. JANICKIEGO 20 B 55-095 MIRKÓW 60-542 POZNAN
NIP 915-151-14-82 NIP 778-10-27-841 BANK : BZWBK 2 O.we

Wrocławiu vvUUUTB4618 _ KONTO 37 1090 2402 0000 0001 1012
0149 Vat Lp. | Nazwatowani lub usługi. Lm. Rar netto | Wartość -
Wartość ru „Alość ——— > netto '<1% Wartość brutto 1 JvŁĄCZNIK
POZAROWW = - | 1 szt 79 7800 79,00j23 18,17 97,17 RAZEM 79,00
18,17 97,17 stawki vat 79,00j23 18,37 97,17 RAZEM : 97,17 zł Słownie
: dziewięćdziesiąt siedem złotych siedemnaście groszy Do zapłaty
97,17 zł przelewem do dnia 15-06-2016 (14 dni od daty wystawienia)
czytelny podpis osoby upoważnionej do odbioru faktur VAT podpis /
pieczęć osoby upoważnionej do wystawiania faktur VAT Krzysztof
Omelko

Lista tokenów:

„OMEGA”, „Faktura”, „VAT”, „nr”, „2/06/16”, „strona”, „1/1”, „Faktura”,
„VAT”, „nr2/06/16”, „Oryginał”, „Miejs”, „:”, „BKUROWIEEFA”,
„SYSTEM”, „|”, „»”, „Data”, „wystawienia”, „:”, „AM”, „|”, „2016”,
„SPRZEDAWCA”, „:”, „”, „:”, „:”, „NABYWCA”, „:”, „—”, „—”, „—”,
„OMEGA”, „SYSTEM”, „KRZYSZTOF”, „OMELKO”, „CLARSYSTEM”,
„Ss”, „:”, „A”, „:”, „PASIKUROWICE”, „UL”, „:”, „LEŚNA”, „7”, „UL”, „:”,
„JANICKIEGO”, „20”, „B”, „55”, „-”, „095”, „MIRKÓW”, „60”, „-”, „542”,
„POZNAN”, „NIP”, „915”, „-”, „151”, „-”, „14”, „-”, „82”, „NIP”, „778”, „-”,
„10”, „-”, „27”, „-”, „841”, „BANK”, „:”, „BZWBK”, „2”, „O.we”,
„Wrocławiu”, „vvUUUTB4618”, „_”, „KONTO”, „37”, „1090”, „2402”,
„0000”, „0001”, „1012”, „0149”, „Vat”, „Lp”, „:”, „|”, „Nazwatowani”,
„lub”, „usługi”, „:”, „Lm”, „:”, „Rar”, „netto”, „|”, „Wartość”, „-”, „Wartość”,
„ru”, „:”, „Alość”, „—”, „—”, „—”, „>”, „netto”, „\”, „<”, „1”, „%”,
„Wartość”, „brutto”, „1”, „JvŁĄCZNIK”, „POZAROWW”, „=”, „-”, „|”, „1”,
„szt”, „79”, „7800”, „79,00j23”, „18,17”, „97,17”, „RAZEM”, „79,00”,
„18,17”, „97,17”, „stawki”, „vat”, „79,00j23”, „18,37”, „97,17”, „RAZEM”,
„:”, „97,17”, „zł”, „Słownie”, „:”, „dziewięćdziesiąt”, „siedem”, „złotych”,
„siedemnaście”, „groszy”, „Do”, „zapłaty”, „97,17”, „zł”, „przelewem”,
„do”, „dnia”, „15”, „-”, „06”, „-”, „2016”, „(”, „14”, „dni”, „od”, „daty”,
„wystawienia”, „)”, „czytelny”, „podpis”, „osoby”, „upoważnionej”, „do”,
„odbioru”, „faktur”, „VAT”, „podpis”, „/”, „pieczęć”, „osoby”,
„upoważnionej”, „do”, „wystawiania”, „faktur”, „VAT”, „Krzysztof”,
„Omelko”

Lista słów kluczowych:

„omega”, „faktura”, „vat”, „2/06/16”, „strona”, „1/1”, „faktura”, „vat”,
„nr2/06/16”, „oryginał”, „miejs”, „bkurowieefa”, „system”, „data”,
„wystawienia”, „sprzedawca”, „nabywca”, „omega”, „system”,
„krzysztof”, „omelko”, „clarsystem”, „pasikurowice”, „leśna”, „7”,
„janickiego”, „mirków”, „542”, „poznani”, „nip”, „151”, „14”, „nip”, „27”,
„841”, „bank”, „bzwbk”, „wrocławiu”, „vvuuutb4618”, „konto”, „vat”,
„nazwatowani”, „usługi”, „rar”, „netto”, „wartość”, „wartość”, „ru”,
„alość”, „1”, „wartość”, „brutto”, „pozaroww”, „1”, „97,17”, „18,17”,
„stawki”, „79,00j23”, „18,37”, „razem”, „złotych”, „groszy”, „zapłaty”,

"przelewem", "dnia", "06", "dni", "daty", "wystawienia", "czytelny",
"podpis", "osoby", "upoważnionej", "odbioru", "faktur", "vat", "podpis",
"pieczęć", "osoby", "upoważnionej", "wystawiania", "faktur", "vat",
"krzysztof", "omelko"

Częstotliwość wystąpień słów kluczowych w dokumencie:

"vat": 1.0, "wartość": 0.6, "omega": 0.4, "faktura": 0.4, "system": 0.4,
"krzysztof": 0.4, "omelko": 0.4, "nip": 0.4, "1": 0.4, "podpis": 0.4,
"osoby": 0.4, "upoważnionej": 0.4, "faktur": 0.4, "2/06/16": 0.2, "strona":
0.2, "1/1": 0.2, "nr2/06/16": 0.2, "oryginał": 0.2, "miejs": 0.2,
"bkurowieefa": 0.2, "data": 0.2, "wystawienia": 0.2, "sprzedawca": 0.2,
"nabywca": 0.2, "clarsystem": 0.2, "pasikowice": 0.2, "leśna": 0.2, "7":
0.2, "janickiego": 0.2, "mirków": 0.2, "542": 0.2, "poznan": 0.2, "151":
0.2, "14": 0.2, "27": 0.2, "841": 0.2, "bank": 0.2, "bzwbk": 0.2,
"wrocławiu": 0.2, "vvuutb4618": 0.2, "konto": 0.2, "nazwatowani": 0.2,
"usługi": 0.2, "rar": 0.2, "netto": 0.2, "ru": 0.2, "ałość": 0.2, "brutto": 0.2,
"pozaroww": 0.2, "97,17": 0.2, "18,17": 0.2, "stawki": 0.2, "79,00|23":
0.2, "18,37": 0.2, "razem": 0.2, "złotych": 0.2, "groszy": 0.2, "zapłaty":
0.2, "przelewem": 0.2, "dnia": 0.2, "06": 0.2, "dni": 0.2, "daty": 0.2,
"wystawienia": 0.2, "czytelny": 0.2, "odbioru": 0.2, "pieczęć": 0.2,
"wystawiania": 0.2

Wagi zdań:

Zdanie	Waga
OMEGA Faktura VAT nr 2/06/16 strona 1/1 Faktura VAT nr2/06/16 Oryginał Miejs : BKUROWIEEFA SYSTEM »	0.8
Data wystawienia : AM! 2016 SPRZEDAWCA : :.* NABYWCA .	0.2
LEŚNA 7 UL.	0.2
JANICKIEGO 20 B 55-095 MIRKÓW 60-542 POZNAN NIP 915-151-14-82 NIP 778-10-27-841 BANK : BZWBK 2 O.we Wrocławiu vvUUUTB4618 _ KONTO 37 1090 2402 0000 0001 1012	1.0
Nazwatowani lub usługi.	0.2

Lm. Rar netto Wartość - Wartość ru „Alość ——— > netto '<1%	1.0
Wartość brutto 1 JvŁĄCZNIK POZAROWW = - 1 szt 79 7800 79,00j23 18,17 97,17 RAZEM 79,00 18,17 97,17 stawki vat 79,00j23 18,37 97,17 RAZEM : 97,17 zł	3.8
Słownie : dziewięćdziesiąt siedem złotych siedemnaście groszy Do zapłaty 97,17 zł przelewem do dnia 15-06-2016 (14 dni od daty wystawienia) czytelny podpis osoby upoważnionej do odbioru faktur VAT podpis / pieczęć osoby upoważnionej do wystawiania faktur VAT Krzysztof Omelko	6.2

Podsumowanie dokumentu:

Słownie : dziewięćdziesiąt siedem złotych siedemnaście groszy Do zapłaty 97,17 zł przelewem do dnia 15-06-2016 (14 dni od daty wystawienia) czytelny podpis osoby upoważnionej do odbioru faktur VAT podpis / pieczęć osoby upoważnionej do wystawiania faktur VAT Krzysztof Omelko

Podsumowanie po usunięciu znaków interpunkcyjnych:

Słownie dziewięćdziesiąt siedem złotych siedemnaście groszy Do zapłaty 9717 zł przelewem do dnia 15062016 (14 dni od daty wystawienia) czytelny podpis osoby upoważnionej do odbioru faktur VAT podpis pieczęć osoby upoważnionej do wystawiania faktur VAT Krzysztof Omelko

Znormalizowane tokeny podsumowania:

"słownie", "dziewięćdziesiąt", "siedem", "złoty", "siedemnaście", "grosz", "zapłata", "9717", "złoty", "przelew", "dzień", "15062016", "(", "14", "dzień", "data", "wystawieć", ")", "czytelny", "podpis", "osoba", "upoważniić", "odbior", "faktura", "vat", "podpis", "pieczęć", "osoba", "upoważniony", "wystawiać", "faktura", "vat", "<persName>

Podsumowanie po normalizacji:

słownie dziewięćdziesiąt siedem złoty siedemnaście grosz zapłata 9717 złoty przelew dzień 15062016 (14 dzień data wystawieć) czytelny podpis osoba upoważniić odbiór faktura vat podpis pieczęć osoba upoważniony wystawiać faktura vat <persName>

3. Klasyfikacja dokumentów

1. Wybór modelu

W celu automatycznego wyboru najlepszego modelu do klasyfikacji dokumentów wykonana została walidacja krzyżowa z grid search'em. Zbiór treningowy został podzielony na 5 losowych podzbiorów, które kolejno były trenowane i walidowane.

Do wektoryzacji danych tekstowych zbadane zostały następujące ekstraktory cech:

- Bag-of-Words (BOW) - konwersja podsumowania na wektor reprezentujący wystąpienie tokenów w podsumowaniu;
- TFIDF – przypisanie wagi tokenom w podsumowaniu na podstawie częstości ich wystąpień w danym podsumowaniu w porównaniu do częstości ich wystąpień w całym zbiorze podsumowań;
- Hashing Vectorizer – konwersja tekstu na wektor przy użyciu funkcji haszującej.

Do klasyfikacji dokumentów na podstawie znormalizowanych podsumowań zbadane zostały następujące modele:

- Regresja Logistyczna
- Stochastyczny spadek gradientu
- Klasyfikator wektorów nośnych

Ponadto zbadane zostały różne wartości parametrów modeli:

Model	Parametr
Logistic Regression	C solver penalty multi_class tol
	solver penalty

	multi_class tol
SGD	loss penalty alpha l1_ratio tol
	loss penalty alpha tol
SVC	C gamma kernel decision_function_shape tol

2. Wyniki dla zbioru danych Korespondencja

1. Trenowanie modelu

Wyniki wszystkich badań modeli do klasyfikacji dokumentów zapisane są w pliku *raport_klasyfikator_korespondencja.xlsx*.

Najlepszy wynik (makro F1 0.6433) uzyskano dla ekstraktora cech BOW, modelu regresji logistycznej i następujących wartości parametrów: {'C': 0.1, 'solver': 'sag', 'penalty': 'l2', 'multi_class': 'ovr', 'tol': 0.01}.

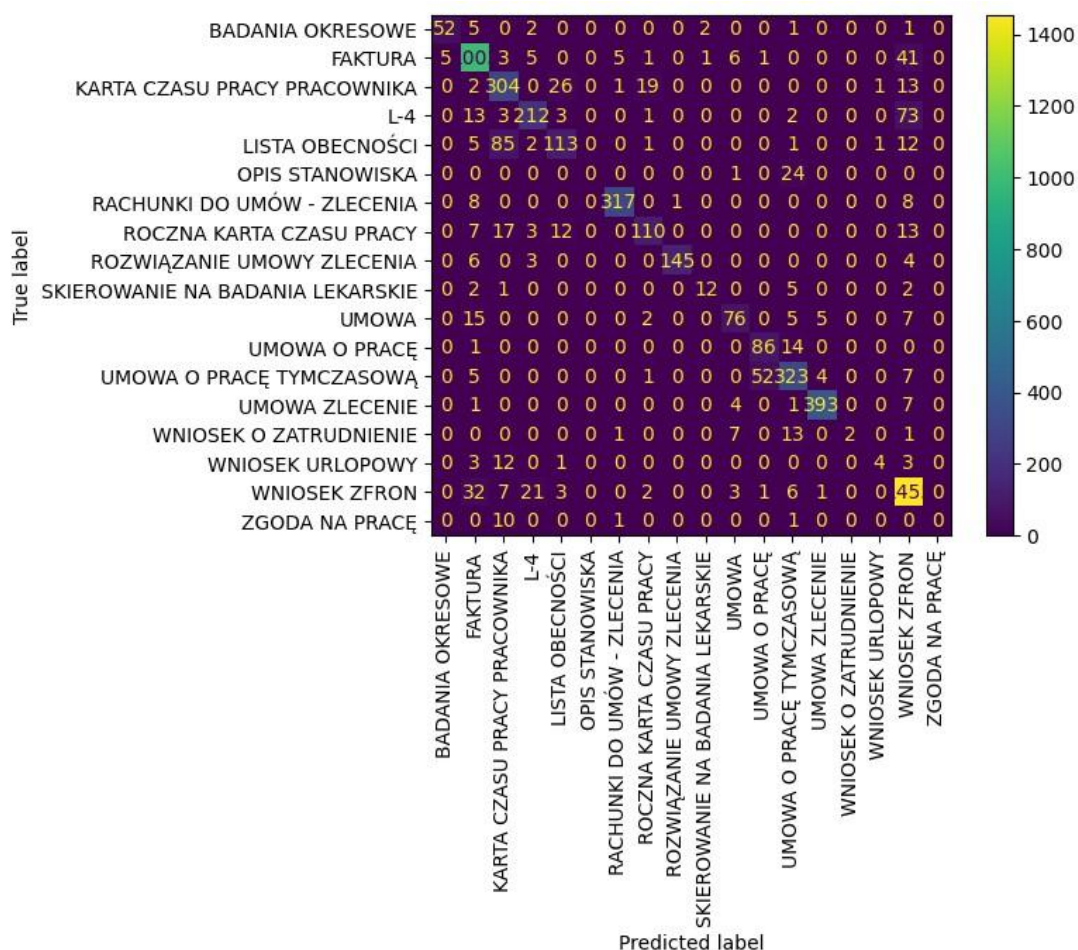
2. Testowanie modelu

Na zbiorze testowym, dla najlepszego modelu, otrzymano następujące wyniki:

RAPORT KLASYFIKACJI

Rodzaj dokumentu	Precyzja	Czułość	F1	Liczba wystąpień
L-4	0.8548	0.6906	0.7640	307
umowa	0.7835	0.6909	0.7343	110
lista obecności	0.7152	0.5136	0.5979	220
opis stanowiska	0.0000	0.0000	0.0000	25
umowa o pracę	0.6143	0.8515	0.7137	101
rachunki do umów - zlecenia	0.9754	0.9491	0.9621	334
karta czasu pracy pracownika	0.6878	0.8306	0.7525	366
zgoda na pracę	0.0000	0.0000	0.0000	12
wniosek urlopowy	0.6667	0.1739	0.2759	23
umowa zlecenie	0.9752	0.9680	0.9716	406
wniosek o zatrudnienie	1.0000	0.0833	0.1538	24
badania okresowe	0.9123	0.8254	0.8667	63
rozwiązanie umowy zlecenia	0.9932	0.9177	0.9539	158
skierowanie na badania lekarskie	0.8000	0.5455	0.6486	22
wniosek zfron	0.8834	0.9504	0.9157	1531
umowa o pracę tymczasową	0.8157	0.8240	0.8198	392
faktura	0.9052	0.9365	0.9206	1071
roczna karta czasu pracy	0.8029	0.6790	0.7358	162
dokładność			0.8648	5327
średnia makro	0.7436	0.6350	0.6548	5327
średnia ważona	0.8606	0.8648	0.8575	5327

MACIERZ KONFUZJI



3. Wyniki dla zbioru danych Korespondencja

1. Trenowanie modelu

Wyniki wszystkich badań modeli do klasyfikacji dokumentów zapisane są w pliku *raport_klasyfikator_umowy100.xlsx*.

Najlepszy wynik (makro F1 0.5658) uzyskano dla ekstraktora cech BOW, modelu stochastyczny spadku gradientu i następujących wartości parametrów: {'loss': 'modified_huber', 'penalty': 'l1', 'alpha': 0.8, 'l1_ratio': 0.007, 'tol': 0.06}.

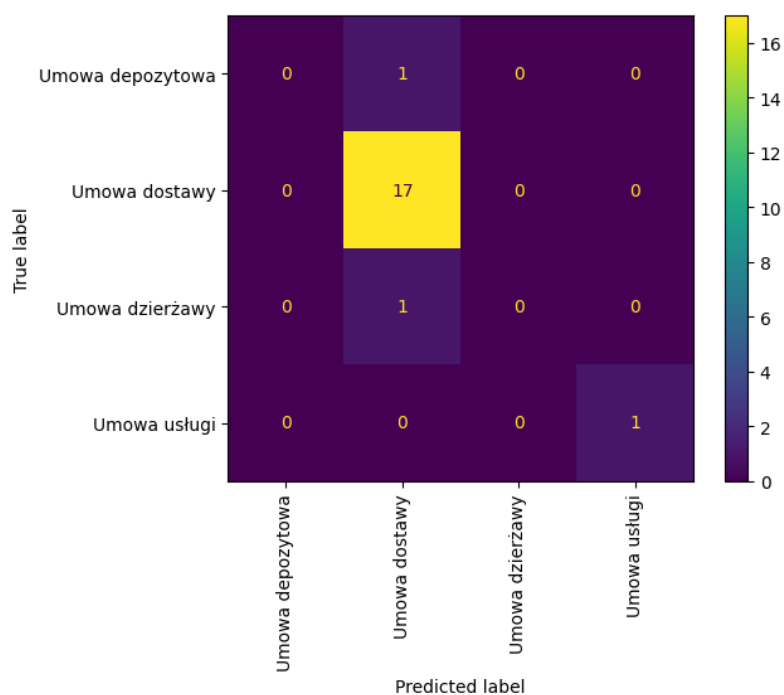
2. Testowanie modelu

Na zbiorze testowym, dla najlepszego modelu, otrzymano następujące wyniki:

RAPORT KLASYFIKACJI

Rodzaj dokumentu	Precyzja	Czułość	F1	Liczba wystąpień
Umowa dostawy	0.8947	1.0000	0.9444	17
Umowa dzierżawy	0.0000	0.0000	0.0000	1
Umowa usługi	1.0000	1.0000	1.0000	1
Umowa depozytowa	0.0000	0.0000	0.0000	1
dokładność			0.9000	20
średnia makro	0.4737	0.5000	0.4861	20
średnia ważona	0.8105	0.9000	0.8528	20

MACIERZ KONFUZJI



4. Parsowanie układu

W zbiorze danych Umowy w wielu dokumentach występuje układ kolumnowy. Poniższy przykład reprezentuje taki układ, gdzie dane sprzedawcy znajdują się po lewej stronie dokumentu, zaś dane nabywcy – po prawej.

SPRZEDAWCA

Firma Krzak

01-120 Warszawa, al. Jerozolimskie 1

NIP: 123-456-78-90 Regon:
123456789

NABYWCA

Anna Nowak

01-320 Bydgoszcz, ul.
Polna 5

PESEL: 81121202678

W wyniku działania algorytmu OCR układ ten zostaje pominięty, w wyniku czego powstaje następujący tekst:

SPRZEDAWCA NABYWCA Firma Krzak Anna Nowak 01-120 Warszawa, al.
Jerozolimskie 1 01-320 Bydgoszcz, ul. Polna 5 NIP: 123-456-78-90 Regon:
123456789 PESEL: 81121202678

Na jego podstawie ekstrakcja danych sprzedawcy oraz danych nabywcy jest niemożliwa.

W celu rozwiązania tego problemu zbadane zostały różne modele do parsowania układu obrazu/dokumentu:

OpenCVModel	SkImageModel	LayoutParser
<ol style="list-style-type: none"> 1. Konwersja obrazu na odcienie szarości 2. Binaryzacja obrazu na podstawie progu Otsu 3. Dylatacja 4. Erozja 5. Znalezienie konturów 	<ol style="list-style-type: none"> 1. Obliczenie średniej wartości każdego piksela (średniej trzech kanałów) 2. Binaryzacja uśrednionego obrazu na podstawie progu Otsu 3. Etykietowanie regionów na podstawie 	<ol style="list-style-type: none"> 1. Konwersja obrazu na BGR 2. Zmiana rozmiaru obrazu według <code>cfg.INPUT.{MIN,MAX}_SIZE_TEST</code> 3. Predykcja układu dokumentu za pomocą pretrenowanego modelu głębokiego

6. Znalezienie bounding box'ów	zbinaryzowanego obrazu 4. Znalezienie bounding box'ów na podstawie etykiet regionów	
--------------------------------	--	--


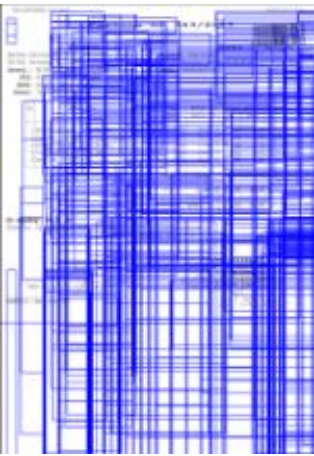
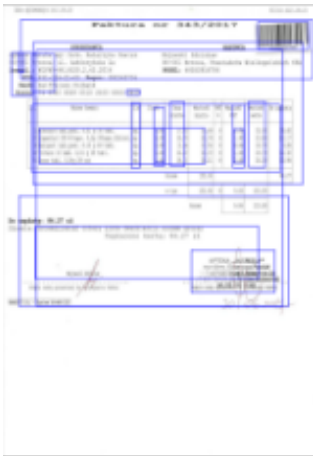
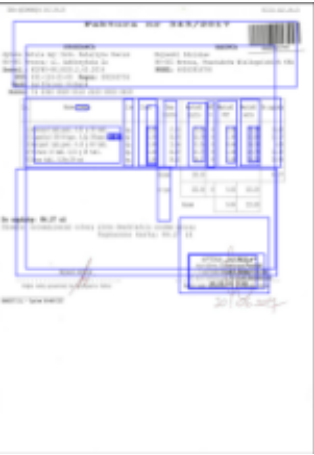


Do modelu LayoutParser przebadane zostały następujące modele:

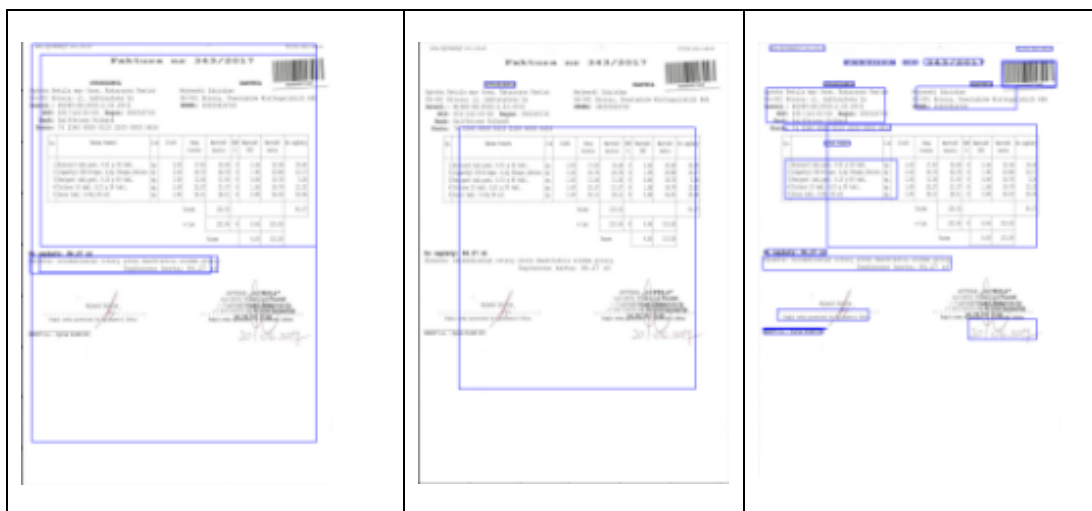
Model	Zbiór danych, na których model został pretrenowany
<u>faster_rcnn_R_50_FPN_3x</u>	<u>HJDataset</u>
<u>mask_rcnn_R_50_FPN_3x</u>	<u>HJDataset</u>
<u>retinanet_R_50_FPN_3x</u>	<u>HJDataset</u>
<u>faster_rcnn_R_50_FPN_3x</u>	<u>PubLayNet</u>
<u>mask_rcnn_R_50_FPN_3x</u>	<u>PubLayNet</u>
<u>mask_rcnn_X_101_32x8d_FPN_3x</u>	<u>PubLayNet</u>
<u>mask_rcnn_R_50_FPN_3x</u>	<u>PrimaLayout</u>

- * W przypadku modeli PubLayNet model mask_rcnn_X_101_32x8d_FPN_3x był trenowany na całym zbiorze treningowym, podczas gdy inne były trenowane tylko na zbiorze walidacyjnym (rozmiar wynosi około 1/50).

Przykładowe wyniki działania parserów układu dokumentów:

OpenCVModel	SkImageModel	LayoutParser (HJDataset & faster_rcnn_R_50_FPN_3x)
-------------	--------------	--

		
LayoutParser (HJDataset & mask_rcnn_R_50_FPN_3x)	LayoutParser (HJDataset & retinanet_R_50_FPN_3x)	LayoutParser (PubLayNet & faster_rcnn_R_50_FPN_3x)
		
LayoutParser (PubLayNet & mask_rcnn_R_50_FPN_3x)	LayoutParser (PubLayNet & mask_rcnn_X_101_32x8d_FPN_3x)	LayoutParser (PrimaLayout & mask_rcnn_R_50_FPN_3x)



Pipeline wykorzystujący parsowanie układu dokumentów:

1. Wczytanie obrazu strony dokumentu.
2. Ekstrakcja bounding box'ów za pomocą jednego z modeli opisanych powyżej.
3. Dla każdego bounding box'a:
 1. Przycięcie obrazu do bounding box'a (z paddingiem).
 2. Detekcja tekstu za pomocą OCR.
4. Połączenie powstałych tekstów w jeden.
5. Ekstrakcja informacji.

PRZYKŁAD dla modelu OpenCVModel

Strona dokumentu:

UMOWA DOSTAWY NR 156-9/D-ZP/PN-M-ZM/2017

zawarta w trybie przetargu nieograniczonego zgodnie z art. 39 Ustawy z dnia 29 stycznia 2004 r. Prawo Zamówień Publicznych zwanej dalej „PZP” (t.j.: Dz. U. 2017 r., poz. 1579)

w dniu 13.03.2018 roku w Gdańsku, pomiędzy **Uniwersyteckim Centrum Klinicznym** z siedzibą w 80-952 Gdańsk, ul. Dębinki 7, działającym zgodnie z wpisem do Krajowego Rejestru Sądowego pod numerem 0000122150, NIP 957-07-30-409, zwanym w dalszej treści umowy „**ZAMAWIAJĄCYM**”, reprezentowanym przez:

Jakub Kraszewski - Dyrektor Naczelny

a

Teleflex Polska Sp. z o.o. z siedzibą: ul. Iłżecka 26/104, 02-135 Warszawa działającą zgodnie z wpisem do Krajowego Rejestru Sądowego, prowadzonego przez Sąd Rejonowy dla M. St. Warszawy w Warszawie, XIII Wydział Gospodarczy KRS pod numerem 0000668462, NIP 5223086403, zwaną w dalszej treści umowy „**WYKONAWCĄ**” dostawy, reprezentowaną przez:

[Podpis] - *[Podpis]*

§ 1

1. Przedmiotem niniejszej umowy jest dostawa jednorazowych i wielorazowych wyrobów medycznych dla UCK w okresie, asortymencie określonych w załączniku nr 1 do niniejszej umowy.
2. Ilość określona w załączniku do niniejszej umowy stanowi wielkość szacunkową i może ulec zmniejszeniu w zależności od potrzeb Zamawiającego.
3. Wartość niniejszej umowy określa się na:
netto: [] PLN (słownie: []),
plus należny podatek VAT

§ 2

1. Dostawy następować będą sukcesywnie, w ilości i asortymencie, zgodnie z zamówieniami częściowymi Zamawiającego w terminie do 2 dni roboczych od dnia otrzymania zamówienia.
2. Zamówienie będzie złożone faksem przez osobę wyznaczoną przez Zamawiającego – Kierownika Działu Zaopatrzenia Medycznego.
3. Dostawa odbędzie się na ryzyko i koszt Wykonawcy do Magazynu Technicznego Działu Zaopatrzenia Medycznego w Gdańsku, ul. [] w dni robocze w godz. 8:00 – 14:00.
4. Odpowiedzialność za dostarczenie przedmiotu zamówienia w terminie i w miejsce wskazane przez Zamawiającego ponosi Wykonawca.
5. Osoba do kontaktu na etapie realizacji umowy po stronie Wykonawcy:

[Podpis] nr tel. 22 462 40 32

Imię i Nazwisko

Adres mailowy do kontaktu z firmą *orders.pl@teleflex.com*

Nr fax do składania zamówień 22 48 53 222

6. Osoby do kontaktu na etapie realizacji umowy po stronie Zamawiającego:

W zakresie realizacji zamówień

Wioletta Kurmin-Sab []

W zakresie obrotu fakturowego

[]

§ 3

1. Wykonawca zobowiązuje się do dostarczenia towaru:
 - a) posiadającego dokumenty dopuszczające przedmiot zamówienia do użytku w placówkach ochrony zdrowia na terenie RP,

1 *[Podpis]*

Przykładowy obraz powstały po przycięciu do	OCR z obrazu	Obraz z paddingiem	OCR z obrazu z paddingiem
---	--------------	--------------------	---------------------------

znalezione bounding box'a			
zobowiązuje	zobowiązuje	zobowiązuje	zobowiązuje
e-mail		e-mail	e-mail
29		29	
349		349	349
fax	fax	fax	fax
umowy	UMOWY	umowy	umowy
stronie		stronie	stronie
etapie	etapie	etapie	etapie
kontaktu		kontaktu	kontaktu
do	do	do	

OCR na obrazie całej strony dokumentu	OCR na
UMOWA DOSTAWY NR 156-9/D-ZP/PN-M-ZM/2017	zgodnie nieograniczonego Usta przetargu stycznia Zamówień Publicznych Zwanej DZ. DOZ. Gdańsku, DOMędzy Uniwersyteckim Klirycznym siedzibą Ul. Dębinki działającym zgodnie do Wpisem Sądowego DOG Krajowego treści dalszej

zawarta w trybie przetargu
nieograniczonego zgodnie z art. 39
Ustawy z dnia 29 stycznia 2004 r.

Prawo Zamówień Publicznych zwanej
dalej „PZP” (t.j.: Dz. U. 2017 r., poz.
1579)

w dniu 13.03.2018 roku w Gdańsku,
pomiędzy Uniwersyteckim Centrum
Klinicznym z siedzibą w

80-952 Gdańsk, ul. Dębinki 7,
działającym zgodnie z wpisem do
Krajowego Rejestru Sądowego pod

numerem 0000122150, NIP 957-07-30-
409, zwanym w dalszej treści umowy
„ZAMAWIAJĄCYM”,

reprezentowanym przez:

Jakub Kraszewski - Dyrektor Naczelny

a

Teleflex Polska Sp. z o.o. z siedzibą: ul.
Iłżecka 26/104, 02-135 Warszawa
działającą zgodnie z

wpisem do Krajowego Rejestru
Sądowego, prowadzonego przez Sąd
Rejonowy dla M. St. Warszawy w

Warszawie, XIII Wydział Gospodarczy
KRS pod numerem 0000668462, NIP
5223086403 , zwaną w

dalszej treści umowy „WYKONAWCĄ”
dostawy, Teprzentowaną przez:

WIAJĄCYM”, LAMA ZWanym
UMOWY reprezentowanym
Kraszewski Dyrektor Naczelny
siedzibą ul. SD. działającą zgodnie
do Sądowego, Wpisem Rejestru
prowadzonego sąd Rejonowy
Wydział Warszawie, Gospodarczy
DOd 0000668462, ZWanaą treści
dalszej dostawy WCA
reprezentowaną UMOWY DrZzeZ.
A (WY wyrobów jednorazowych
niniejszej wielorazowych
medycznych UMOWY określonych
okresie, asortymencie załączniku
do niniejszej UMOWY określona
Wielkość załączniku do niniejszej
szacunkową ulec UMOWY
zależności od zmniejszeniu potrzeb
Zamawiającego. Wartosc określa
niniejszej UmoOwWY
TRZYDZIEŚCI SZEŚĆSET N
TYSIĄCE należy Dodatek ilości
Dostawy sukcesywnie,
zamówieniami zgodnie
asortymencie, częściowymi do dni
zamowienia. Zamawiającego
roboczych 'e'e otrzymania
Zamowienie będzie Złożone
Zamawiającego Drzez
Medycznego. odbędzie
Wykonawcy do echnicznego
Magazynu Gdańsku, Medycznego
atrzenia Ul. Dębinki dni
Odpowiedzialność zamówienia
przedmiotu miejsce
Zamawiającego Wykonawca.
DONOSI r FY/sPY TTP TF etapie
Wykonawcy: 23 U : ces sly na
€Lapi€ I UMOWY Je mailowy do
zamówień fax do Nr Osoby do
realizacji etapie Zamawiającego
UMOWY zamówień realizacji
Kurmin-Sabady fax c-mail
WkurminGQuck.gqda.pl zakresie
fakturowego Łeppek Wykonawca

<p>\$1</p> <p>1. Przedmiotem niniejszej umowy jest dostawa jednorazowych i wielorazowych wyrobów medycznych</p> <p>dla UCK w okresie, asortymencie określonych w załączniku nr 1 do niniejszej umowy.</p> <p>2. Ilość określona w załączniku do niniejszej umowy stanowi wielkość szacunkową i może ulec</p> <p>zmniejszeniu w zależności od potrzeb Zamawiającego.</p> <p>3. Wartość niniejszej umowy określa się na:</p> <p>netto: ----- PLN (słownie: ----- -);</p> <p>plus należny podatek VAT</p> <p>82</p> <p>1. Dostawy następować będą sukcesywnie, w ilości i asortymencie, zgodnie z zamówieniami</p> <p>częściowymi Zamawiającego w terminie do 2 dni roboczych od dnia otrzymania zamówienia.</p> <p>2. Zamówienie będzie złożone faksem przez osobę wyznaczoną przez Zamawiającego — Kierownika</p>	<p>zobowiązuje do posiadającego dokumenty zamówienia dopuszczające przedmiot placówek do ochrony</p>
--	--

Działu Zaopatrzenia Medycznego.

3. Dostawa odbędzie się na ryzyko i koszt Wykonawcy do Magazynu Technicznego Działu

Zaopatrzenia Medycznego w Gdańsku, ul. Dębinki 7 w dni robocze w godz. 8:00 — 14:00.

4. Odpowiedzialność za dostarczenie przedmiotu zamówienia w terminie i w miejsce wskazane przez

a ponosi Wykonawca.

5. O do kontaktu na etapie o nowego po stronie Wykonawcy

srrAAe CE |: Ua OUCOOICU Senne nr tel. . AUD, I. UK) 29a

u Imię i Nazwisko

Adres mailowy do kontaktu z firm m QOS 320 1 A (2 A kc . CO

Nr fax do składania zamówieńC. Ć UE sz Joel pianinie

6. Osoby do kontaktu na etapie realizacji umowy po stronie Zamawiającego:

W zakresie realizacji zamówień

<p>W zakresie obrotu fakturowego</p> <p>-----</p> <p>83</p> <p>1. Wykonawca zobowiązuje się do dostarczenia towaru:</p> <p>a) posiadającego dokumenty dopuszczające przedmiot zamówienia do użytku w placówkach</p> <p>ochrony zdrowia na terenie RP,</p> <p>1 w”</p>	
---	--

Informacje wyekstrahowane z OCR na obrazie całej strony dokumentu	Informacje wyekstrahowane z OCR na
<p>city:Gdańsk, city:Warszawa, document_number:156-9/D-ZP/PN-M-ZM/01, document_number:tel./fax, document_number:.....C, document_number_undefined:004, document_number_undefined:13.03.018, document_number_undefined:14:00, document_number_undefined:159, document_number_undefined:33.600,00, document_number_undefined:349, document_number_undefined:349, document_number_undefined:349, document_number_undefined:53086403, document_number_undefined:8:00, document_number_undefined:o.o, document_number_undefined:t.j., num_days:2, num_days:7, phone_num:000066846, province:mazowieckie, province:pomorskie, street number:26,</p>	<p>nip:0000668462</p>

street:Centrum, street:Ilżecka, zip_code:02-135, zip_code:80-952	
---	--

5. Ekstrakcja informacji

1. Statystyki

Plik *raport_statystyki_dla_ekstraktora.xlsx* przedstawia dwie tabele – dla obu zbiorów danych. W każdej z nich jest informacja o:

- Liczbie wystąpień metadanych każdego typu w treści zOCRowanego dokumentu, z podziałem na rodzaje dokumentów.
- Liczbie i procentowej wartości dokładnych dopasowani metadanych w treści.
- Liczbie i procentowej wartości podobnych dopasowani metadanych w treści. Podobne dopasowania obliczone zostały na podstawie odległości Levenshteina z progiem odcinającym równym 0.7 znaków w słowie.

2. Modele do ekstrakcji:

1. Modele heurystyczne/regułowe

- nip
 - Liczba o 10 cyfrach.
 - Ostatnia cyfra równa sumie iloczynu cyfr przez następujące wagi: 6, 5, 7, 2, 3, 4, 5, 6, 7.
- zipCode
 - Dwie cyfry + możliwość wystąpienia spacji + myślnik + możliwość wystąpienia spacji + trzy cyfry.
- Kwoty pieniężne: netValue, grossValue
 - Liczba (z możliwymi spacjami, kropkami i/lub przecinkami pomiędzy cyframi) + jedno ze słów: „zł”, „PLN”, „gr” z pominięciem takich wyszukiwań, które poprzedzone są słowem „słownie”.
- proceedingsNumber
 - Od jednej do czterech cyfr + string „/PN/” + rok.
- house
 - Numer występujący po znalezionej nazwie ulicy.

- Daty – startDate, endDate, signDate
 - Formaty:
 - D.M.RR, D.M.RRRR, D.MM.RR, D.MM.RRRR, DD.M.RR, DD.MM.RRRR, DD.MM.RR, DD.MM.RRRR
 - D-M-RR, D-M-RRRR, D-MM-RR, D-MM-RRRR, DD-M-RR, DD-MM-RRRR, DD-MM-RR, DD-MM-RRRR
 - D/M/RR, D/M/RRRR, D/MM/RR, D/MM/RRRR, DD/M/RR, DD/MM/RRRR, DD/MM/RR, DD/MM/RRRR
- number (numer dokumentu)
 - Dopasowanie ciągów znaków bez uwzględniania wielkości liter, które:
 - Nie są poprzedzone przez znaki alfanumeryczne, ukośnik, kropkę lub przecinek.
 - Złożone są z:
 - Co najmniej jednego znaku alfanumerycznego lub znaku interpunkcyjnego;
 - Następnie pojedynczej cyfry lub znaku interpunkcyjnego;
 - Następnie co najmniej jednego znaku alfanumerycznego, ukośnika, myślnika lub odwróconego ukośnika.

2. Modele oparte na odległości Levenshtein'a

- street
 - Przeszukiwanie słów w odległości 5 w przód i 5 w tył od kodu pocztowego.
 - Porównywanie znalezionych słów z nazwami ulic z wcześniej stworzonej listy i wybranie takiego o najlepszym dopasowaniu.

3. Modele słownikowe

- Adres: province, city
 - Słownik kodów pocztowych z odpowiadającymi im nazwami województw oraz miast.

4. dla: ulica, daty, kwoty brutto i netto, liczba dni, nazwa firmy. Dla każdego rodzaju encji powstała lista kandydatów do porównywania.

5. SpaCy

6. PolDeepNER2

- name (nazwa firmy)
- Model HerBERT pretrenowany dla zadania NER z następującymi etykietami: HerBERT Large Cased model pretrained for the task of Named Entity Recognition with the following labels: `nam_adj`, `nam_adj_city`, `nam_adj_country`, `nam_adj_person`, `nam_eve`, `nam_eve_human`, `nam_eve_human_cultural`, `nam_eve_human_holiday`, `nam_eve_human_sport`, `nam_fac_bridge`, `nam_fac_goe`, `nam_fac_goe_stop`, `nam_fac_park`, `nam_fac_road`, `nam_fac_square`, `nam_fac_system`, `nam_liv_animal`, `nam_liv_character`, `nam_liv_god`, `nam_liv_habitant`, `nam_liv_person`, `nam_loc`, `nam_loc_astronomical`, `nam_loc_country_region`, `nam_loc_gpe_admin1`, `nam_loc_gpe_admin2`, `nam_loc_gpe_admin3`, `nam_loc_gpe_city`, `nam_loc_gpe_conurbation`, `nam_loc_gpe_country`, `nam_loc_gpe_district`, `nam_loc_gpe_subdivision`, `nam_loc_historical_region`, `nam_loc_hydronym`, `nam_loc_hydronym_lake`, `nam_loc_hydronym_ocean`, `nam_loc_hydronym_river`, `nam_loc_hydronym_sea`, `nam_loc_land`, `nam_loc_land_continent`, `nam_loc_land_island`, `nam_loc_land_mountain`, `nam_loc_land_peak`, `nam_loc_land_region`, `nam_num_house`, `nam_num_phone`, `nam_org_company`, `nam_org_group`, `nam_org_group_band`, `nam_org_group_team`, `nam_org_institution`, `nam_org_nation`, `nam_org_organization`, `nam_org_organization_sub`, `nam_org_political_party`, `nam_oth`, `nam_oth_currency`, `nam_oth_data_format`, `nam_oth_license`, `nam_oth_position`, `nam_oth_tech`, `nam_oth_www`, `nam_pro`, `nam_pro_award`, `nam_pro_brand`, `nam_pro_media`, `nam_pro_media_periodic`, `nam_pro_media_radio`, `nam_pro_media_tv`, `nam_pro_media_web`, `nam_pro_model_car`, `nam_pro_software`, `nam_pro_software_game`, `nam_pro_title`, `nam_pro_title_album`, `nam_pro_title_article`, `nam_pro_title_book`, `nam_pro_title_document`, `nam_pro_title_song`, `nam_pro_title_treaty`, `nam_pro_title_tv`, `nam_pro_vehicle`,
- Mapowanie etykiet według słownika

<code>nam_org_company</code>	<code>name</code>
<code>nam_org_institution</code>	<code>name</code>

nam_org_organization	name
nam_org_organization_s	
ub	name

3. Metryki

Dla każdego rodzaju encji zostały obliczone następujące metryki:

- liczba encji z prawidłowym tekstem (correct)
- liczba encji pominiętych – nie znalezionych (missed)
- precyzja
- recall
- F1
- liczba wszystkich znalezionych encji (actual)
- liczba wszystkich właściwych encji – z metadanych (possible)

4. Wyniki

1. Zbiór danych Umowy 100

Wyniki znajdują się w pliku *raport_ekstraktor_umowy100.xlsx*.

6. Klasyfikacja wyekstrahowanych informacji

Dla modeli ekstraktorów wyszukiwane były encje o kategoriach szerszych od rodzajów encji w zbiorze danych. Przykładowo dla dat w zbiorze występują 3 kategorie: startDate, endDate, signDate. Z tego względu nawet w przypadku wysokiej wartości recall, precyzja jest niska. Mając na celu poprawę wyników,

7. Pipeline

TRENING

1. Wstępne przetwarzanie dokumentów

Ekstrakcja i wstępne przetwarzanie podsumowania dokumentu (opisane w sekcji 2.1).

2. Dwustopniowy wybór najlepszego modelu do klasyfikacji makr

Opisany w punkcie 3.1. Zbiór treningowy dzielony jest na 5 części, po czym 5 razy przeprowadzany jest trening i ewaluacja – za każdym razem dla innej 1/5 zbioru wykorzystanego do ewaluacji.

W pierwszym etapie brane są pod uwagę elementy 2.1, 2.2, 2.3, 2.4. Trenowane są wszystkie możliwe kombinacje tych elementów i wybierana jest taka, dla której makro F1 ma najwyższą wartość. Tutaj, liczbowe parametry klasyfikatorów (2.4) mają wartości znacznie różniące się.

W drugim etapie, dla wybranych elementów, badane są liczbowe parametry modelu (2.4) z mniejszymi różnicami pomiędzy ich wartościami. W ten sposób znajdowane są ostateczne wartości parametrów modelu.

Szczegóły na temat poszczególnych elementów:

2.1 Ekstraktor cech jest modelem służącym do wektoryzacji danych wejściowych (czyli konwersji zbioru dokumentów tekstowych na macierz wystąpień tokenów).

Testowane ekstraktory cech:

- BoW (Bag of Words)
- TFIDF (Term Frequency-Inverse Document Frequency)
- Hash

2.2 Klasyfikator makr

Testowane modele:

- LogReg - regresja logistyczna
- SGD (Stochastic Gradient Descent)
- SVC (Support Vector Classifier)

2.3 Parametry klasyfikatora

Dla każdego klasyfikatora optymalizowane są odpowiadające mu parametry.

3. Trening najlepszego modelu do klasyfikacji

Trening (na całym zbiorze danych treningowych) z wykorzystaniem elementów wybranych w kroku poprzednim (sekcja 3.2).

4. Trening modeli do ekstrakcji informacji

5. Ekstrakcja informacji (encji)

6. Wybór modelu do klasyfikacji encji

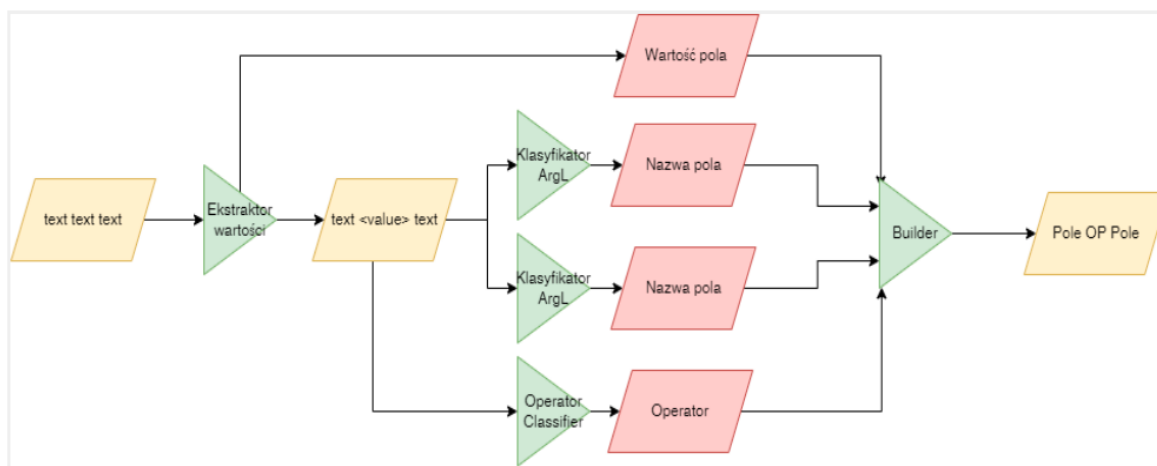
Taki sam sposób jak w przypadku wyboru modelu do klasyfikacji dokumentów. Dane wejściowe to encje (ekstrahowane w kroku poprzednim) wraz ze słowami je otaczającymi.

7. Trening modelu do klasyfikacji encji

Należy tutaj nadmienić, że jako wynik projektu należy zakwalifikować nie tylko uzyskane konkretne wyniki eksperymentów, ale innowacyjna i wartościowa jest opracowana architektura i metodyka, która pozwala zastosować opisane podejście i stworzone w projekcie narzędzia

do rozwiązywania tego typu problemów w przyszłości dla różnych dziedzin i rodzajów dokumentów.

Opis wyniku działania ewaluatora reguł i pozyskana nowa wiedza:



Z przyjętej architektury rozwiązania wynikało, że główne komponenty rozwiązania zostały zidentyfikowane jako klasyfikatory, które prognozują na podstawie fragmentu tekstu:

1. Rodzaj operatora z puli zdefiniowanych w ontologii operatorów;
2. Nazwę pola argumentu lewego operatora z puli zdefiniowanych w ontologii pól;
3. Nazwę pola argumentu prawego operatora z puli zdefiniowanych w ontologii pól.

Nazwa prawego argumentu operatora może być zastąpiona wartością literalnie określoną w tekście. Wartość ta jest wyluskiwana z tekstu przez ekstraktor wartości.

W wyniku eksperymentów uzyskano następujące najlepsze wyniki dla poszczególnych klasyfikatorów:

klasyfikator	makro F1	ekstraktor cech	model do klasyfikacji
operator	0.821	HASH	LogReg
ArgL	0.741	BOW	SGD
ArgR	0.683	BOW	LogReg

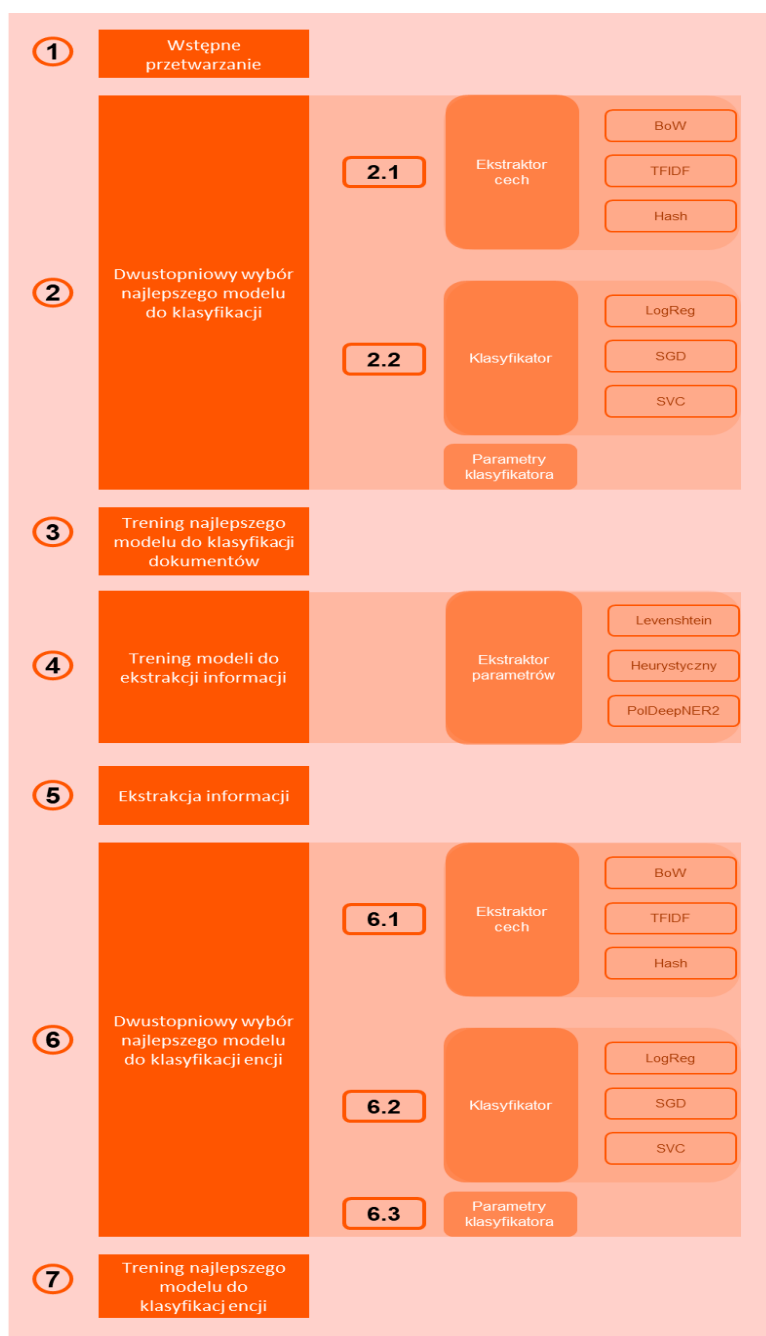
Potwierdzono zatem osiągnięcie kamienia milowego dla Zadania 1 - Opracowanie metod przetwarzania tekstu pod kątem ekstrakcji wyrażeń reprezentujących reguły biznesowe:

- algorytm zdolny do ekstrakcji kluczowych wartości z korpusu 1000 dokumentów z poprawnością identyfikacji na poziomie 80,00 % co najmniej dla fraz definiujących:

kluczowe terminy umowne, wartości kwot wynikających w umów lub innych dokumentów, terminów płatności wynikających z umów / innych dokumentów.

KAMIEŃ MIŁOWY:

- Uzyskano algorytm zdolny do ekstrakcji kluczowych wartości z korpusu 1000 dokumentów z poprawnością identyfikacji na poziomie 82,10 % co najmniej dla fraz definiujących: kluczowe terminy umowne, wartości kwot wynikających w umów lub innych dokumentów, terminów płatności wynikających z umów / innych dokumentów.



Działanie 2 (badania przemysłowe) - Opracowanie metod translacji reguł biznesowych wyrażonych w tekście na reguły wykonywalne, opracowanie silnika ewaluacji reguł.

W ramach niniejszego etapu zrealizowano 4 zadania:

1. Opracowanie algorytmu wiązania dokumentów w kontekst biznesowy na podstawie treści dokumentów

Nowy dokument przychodzący jest zaczątkiem kontekstu biznesowego. System powinien automatycznie uzupełnić kontekst o pozostałe dokumenty. Opracowano dwa algorytmy pierwszy z nich służyć będzie do uzupełniania kontekstu na podstawie równości tożsamościowej encji zawartych w dokumentach (kontrahent, data, numer), drugi zaś, do uzupełniania kontekstu na podstawie treści dokumentów przez opracowanie odpowiedniej miary podobieństwa treści dokumentów.

Szczegółowo opracowane zostały dwa algorytmy, pierwszy z nich służy do uzupełniania kontekstu na podstawie równości tożsamościowej encji zawartych w dokumentach (kontrahent, data, numer), drugi zaś, do uzupełniania kontekstu na podstawie treści dokumentów przez opracowanie odpowiedniej miary podobieństwa treści dokumentów.

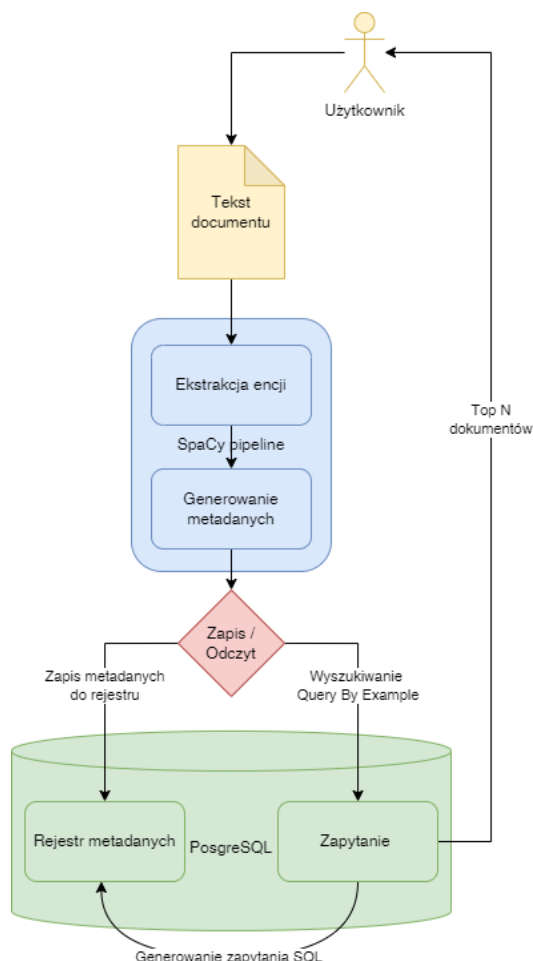
Algorytm wyszukiwania na podstawie tożsamości encji:

Główne założenie algorytmu wyszukiwania na podstawie tożsamości encji zakłada, że dla danego dokumentu wejściowego wyszukiujemy w rejestrze dokumentów takie dokumenty, w których występują te same wartości metadanych. Jest to rodzaj wyszukiwania Query By Example, gdzie wartości metadanych są wejściem do algorytmu generującego zapytanie do bazy danych.

Działanie algorytmu składa się z następujących kroków:

1. Dokument wejściowy pojawiający się w skrzynce wejściowej jest wstępnie przetworzony zgodnie z zaprojektowanym wcześniej procesem.
2. Dokument jest klasyfikowany w celu skierowania na odpowiednią ścieżkę ekstrakcji danych.
3. Z dokumentu są ekstrahowane metadane za pomocą wcześniej opracowanych algorytmów specyficznych dla danego rodzaju dokumentów. Na podstawie wyekstrahowanych metadanych generowany jest dokument JSON.
4. Jeśli dokument ma być zapisany jako nowy dokument w rejestrze to jest on zapisywany w odpowiedniej tabeli bazy danych.
5. Jeśli mają być pobrane z rejestru dokumenty powiązane to na podstawie dokumentu JSON generowane jest zapytanie SQL z odpowiednią klauzulą, która wybiera pasujące

dokumenty. Wybrane dokumenty (ich metadane) są dołączane do kontekstu przetwarzania procesu biznesowego.



Powyższy algorytm zakłada, że w kontekście przetwarzania powinny znaleźć się dokumenty, które są ze sobą logicznie powiązane poprzez wystąpienie tych samych wartości metadanych. W praktyce dla każdego rodzaju dokumentu (umowa, faktura, korespondencja etc.) konieczne jest określenie podzbioru tzw. atrybutów korelujących. Są to najczęściej atrybuty słownikowe / wyliczeniowe a pomijane są atrybuty skalarne, takie jak kwoty, daty etc. ponieważ ich uwzględnienie spowodowałoby występowanie tzw. false positives (np. ta sama wartość daty może występować w wielu niezwiązanych logicznie dokumentach). Na przykład dla wiązania faktur z umowami głównymi atrybutami korelującymi są NIP kontrahenta i numer umowy. Dla wiązania korespondencji w celu tworzenia logicznych “paczek” powiązanych dokumentów jest to głównie nazwa nadawcy. Atrybut daty zarejestrowania dokumentu raczej nie jest traktowany jako atrybut korelujący ale może być użyty do ograniczania ilości powiązanych dokumentów (na przykład interesuje nas korespondencja z okresu roku). Aby była możliwa automatyzacja takiego wiązania dokumentów konieczne było opracowanie koncepcyjne macierzy korelacji atrybutów. Macierz taka obejmuje:

1. Nazwę klasy wejściowego dokumentu.

2. Nazwę klasy dokumentu poszukiwanego,
3. Nazwę atrybutu dokument wejściowego.
4. Nazwę atrybutu dokumentu poszukiwanego
5. Operator porównywania atrybutów (może to być przyrównanie bezpośrednie dla atrybutów wyliczeniowych lub słownikowych, podobieństwo tekstowe na podstawie np. odległości Levenshteina dla atrybutów tekstowych nie słownikowych, zawieranie w przedziale +/- dla dat).

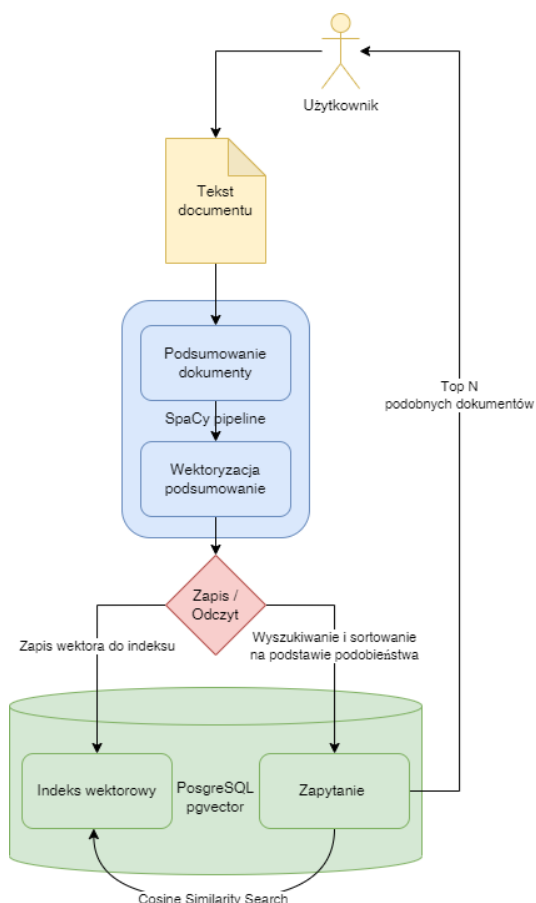
Powyższy algorytm daje wyniki poprawne na poziomie 70 do 80% poprawności, gdzie jako przypadek poprawny bierzmy pod uwagę odnalezienie spodziewanego dokumentu. Nieodnalezienie spodziewanego dokumentu lub dołączenie dokumentu nieprawidłowego traktowane jest jako błąd. Weryfikacja poprawności działania odbywała się poprzez ręczną weryfikację i ocenę użytkowników. Główną przyczyną błędów są problemy związane z brakiem standaryzacji zapisu wartości atrybutów np. nazw własnych lub błędy w ekstrakcji danych.

Algorytm wyszukiwania na podstawie podobieństwa dokumentów:

Alternatywnym podejściem do wyszukiwania dokumentów powiązanych z danym dokumentem wyjściowym jest przeszukiwanie rejestru istniejących dokumentów i dodanie do kontekstu przetwarzania dokumentów, których treść (zawartość tekstowa) jest "podobna" do danego dokumentu. Problem ten można potraktować jako przypadek analizy podobieństwa dokumentów. Dla tego problemu jest możliwe opracowanie potoku przetwarzania tekstu przy wykorzystaniu biblioteki SpaCy. Jako miarę podobieństwa przyjmuje się rodzaj odległości wektorowej, najczęściej odległość cosinusową (cosine similarity). Ze względu na zmienną długość dokumentu należy test dokumentu podsumować do wektora o stałej długości. Dla takich wektorów możliwe jest obliczenie odległości, w tym odległości cosinusowej. Główny problem zastosowania do tego celu biblioteki SpaCy jest związany z wydajnością takiego rozwiązania. Przy dużej bazie dokumentów (dziesiątki tysięcy) problematyczne jest przechowywanie wektorów podsumowań czy to w pamięci czy to w postaci plików na dysku. W takim rozwiązaniu konieczne byłoby również iterowanie po całej liście dokumentów w celu wyszukania dokumentów podobnych (wektorów najbliższych). Dlatego konieczne było opracowanie innego rozwiązania, które byłoby efektywne przy dużej i rosnącej w czasie bazie dokumentów. Rozwiązaniem tego problemu jest przechowywanie dokumentów w wektorowej bazie danych. Baza ta umożliwia przechowywanie wektorów w sposób zoptymalizowany z **możliwością** utworzenia indeksu, który **umożliwia** szybsze przeszukiwanie przestrzeni wektorowej przez hashowanie wartości wektorów, podział całej przestrzeni wektorowej na klastry i redukcję wymiarów. W projekcie wykorzystano jako bazę wektorową moduł pgvector bazy danych PostgreSQL. Baza danych implementuje szybkie przeszukiwanie tak powstałego indeksu pod kątem np. odległości cosinusowej. Ciekawym aspektem rozwiązania jest fakt, że sama analiza **odległości** wektorów jest niezależna od sposobu generacji (embeddingu) tych wektorów. Możliwe jest zarówno stosowanie embeddingu word2vec jak i prostszych metod takich jak TF-IDF czy Bag Of Words.

Działanie algorytmu składa się z następujących kroków:

1. Dokument wejściowy pojawiający się w skrzynce wejściowej jest wstępnie przetworzony zgodnie z zaprojektowanym wcześniej procesem.
2. Treść dokumentu podlega sumaryzacji ekstraktywnej, to jest na podstawie statystyki wystąpienia słów kluczowych (po uprzednim usunięciu "stop words" i nieistotnych części mowy jak zaimki) wyznaczane są zdania, które zawierają najbardziej znaczące słowa kluczowe.
3. Podsumowanie podlega wektoryzacji, to jest poszczególne słowa są kodowane zgodnie z przyjętym modelem. W projekcie jako podstawowy model wektoryzacji przyjęto BOW.
4. Jeśli do korpusu dokumentów ma zostać dodany nowy dokument to jest on zapisywany w bazie w postaci wektorowej.
5. Jeśli mają być pobrane z rejestru dokumenty powiązane to generowane jest zapytanie do bazy wektorowej, które zwraca top N dokumentów posortowanych według cosinusowej miary podobieństwa.



Powyższy algorytm daje wyniki poprawne na poziomie 30 do 60% poprawności, gdzie jako przypadek poprawny bierzmy pod uwagę odnalezienie spodziewanego dokumentu.

Nieodnalezienie spodziewanego dokumentu lub dołączenie dokumentu nieprawidłowego traktowane jest jako błąd. Weryfikacja poprawności działania odbywała się poprzez ręczną weryfikację i ocenę użytkowników. Uzyskane dane pokazały, że bazowanie na podobieństwie wektorowym treści dokumentów nie dało satysfakcjonujących efektów. Jest to znany problem związany z interpretacją biznesowa analizy podobieństwa. Np. zdania “lubię pizze” i “nie lubię kebabu” są podobne jeśli bierzemy pod uwagę odniesienie do jedzenia ale jeśli analizujemy sentyment to oba zdania należą do zupełnie skrajnych klas (pozytywny i negatywny sentyment). Ten sam problem dotyczy dokumentów biznesowych. Na przykład w wyniku przeprowadzonych badań zaobserwowano, że faktura, która jest wystawiana zgodnie z wcześniej podpisana umową może zawierać niewiele wspólnej treści z odpowiedni aumową, która jest też dokumentem znacznie dłuższym, co utrudnia podsumowywanie. W praktyce słowa, które są wspólne dla obu dokumentów dotyczą głównie identyfikacji podmiotów (kupującego i sprzedającego), co de facto sprowadza działanie algorytmu do wykrywania tożsamyh encji. Stosowanie probabilistycznej oceny podobieństwa niesie również ryzyko związane z trudnością oceny istotności wartości podobieństwa i doboru progu istotności. Może to skutkować albo odrzucaniem istotnych dokumentów (false negative) albo zbyt duża ilośća dokumentów w kontekście (false positive).

Na podstawie przeprowadzonych obserwacji przyjęto, że w projekcie jako główna metoda łączenia dokumentów stosowany będzie algorytm bazujący na tożsamości encji.

Przebieg prac badawczych:

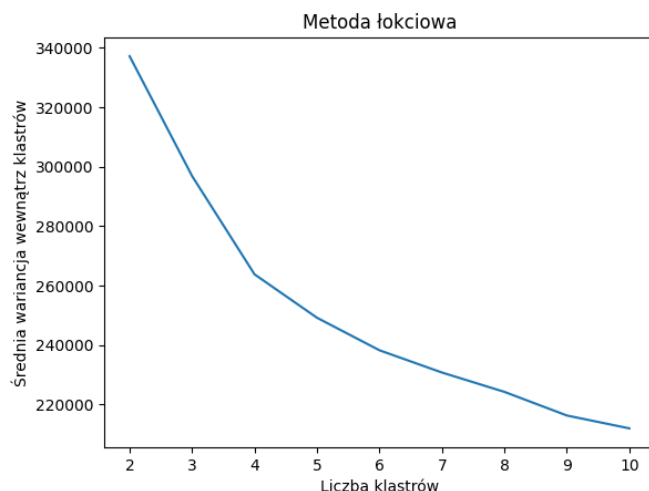
W ramach realizacji tego zadania jako pierwszy problem wzięto pod uwagę możliwość wykorzystania do łączenia dokumentów analizy podobieństwa. Aby zweryfikować zachowanie algorytmów bazujących na podobieństwie najpierw przeprowadzono eksperyment polegający na wektoryzacji zbioru danych różnymi metodami wektoryzacji, aby następnie zbadać możliwość automatycznego porządkowania dokumentów w klastry za pomocą uczenia nienadzorowanego. Krok ten miał dwa cele:

1. Sprawdzenie jakości wektoryzacji testowanego zbioru danych dla różnych metod wektoryzacji uwzględniających różne rodzaje algorytmów wektoryzacji (TFIDF, BOW, FastText, word2vec) oraz różny sposób klasteryzacji (dbscan, kmeans dla różnych parametrów).
2. Przyspieszenie algorytmu szukania podobnych dokumentów na podstawie odległości wektorowej poprzez wstępną klasyfikację dokumentu poprzez przypisanie wartości wektorowej dokumentu do danego klastra, a w następnej kolejności wyszukiwanie najbardziej zbliżonych dokumentów już w danym klastrze.

Jako miarę odległości przyjęto podobieństwo cosinusowe:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

Dla podstawowego zbioru danych przy założeniu wektoryzacji całego dokumentu wyznaczono metodą łokciową optymalną ilość klastrow na 5:



Klasteryzacja zarówno za pomocą metod dbscan jak i kmeans pokazała, że zgodnie z oczekiwaniami średnie podobieństwo pomiędzy dokumentami pochodzącymi z tego samego klastra jest większe niż między dokumentami pochodzącymi z różnych klastrow, co pokazuje poniższy wydruk analizy macierzy średniego podobieństwa między klastrami, gdzie najwyższe wartości występują na przekątnej macierzy:

```
pprint(kmeans_similarities)
```

```
{'cluster 0 - cluster 0': 0.8966921587803127,
 'cluster 0 - cluster 1': 0.5930859380641034,
 'cluster 0 - cluster 2': 0.682358618991823,
 'cluster 0 - cluster 3': 0.6974399658406584,
 'cluster 0 - cluster 4': 0.6205624217316399,
 'cluster 1 - cluster 0': 0.5930859380641035,
 'cluster 1 - cluster 1': 0.8940173347459673,
 'cluster 1 - cluster 2': 0.8218124138962751,
 'cluster 1 - cluster 3': 0.6430648350358344,
 'cluster 1 - cluster 4': 0.38241280967270147,
 'cluster 2 - cluster 0': 0.6823586189918237,
 'cluster 2 - cluster 1': 0.8218124138962738,
 'cluster 2 - cluster 2': 0.8739233797551494,
 'cluster 2 - cluster 3': 0.796758139601643,
 'cluster 2 - cluster 4': 0.591339785690613,
 'cluster 3 - cluster 0': 0.6974399658406581,
 'cluster 3 - cluster 1': 0.6430648350358343,
 'cluster 3 - cluster 2': 0.7967581396016429,
```

```
'cluster 3 - cluster 3': 0.8396645371943748,
'cluster 3 - cluster 4': 0.7397103174832047,
'cluster 4 - cluster 0': 0.6205624217316393,
'cluster 4 - cluster 1': 0.38241280967270175,
'cluster 4 - cluster 2': 0.5913397856906135,
'cluster 4 - cluster 3': 0.739710317483204,
'cluster 4 - cluster 4': 0.8091601432916039}
```

```
pprint(dbscan_similarities)
```

```
{'cluster 0 - cluster 0': 0.5483734641657813,
'cluster 0 - cluster 1': 0.5287662022803508,
'cluster 0 - cluster 2': 0.1111716532098573,
'cluster 0 - cluster 3': 0.6253369199550588,
'cluster 0 - cluster 4': 0.5972522450202912,
'cluster 1 - cluster 0': 0.5287662022803509,
'cluster 1 - cluster 1': 0.7439015072960241,
'cluster 1 - cluster 2': 0.09121719322996534,
'cluster 1 - cluster 3': 0.5018818999581538,
'cluster 1 - cluster 4': 0.48060806389376126,
'cluster 2 - cluster 0': 0.1111716532098573,
'cluster 2 - cluster 1': 0.09121719322996537,
'cluster 2 - cluster 2': 1.0,
'cluster 2 - cluster 3': 0.10304450050414354,
'cluster 2 - cluster 4': 0.028939591901842553,
'cluster 3 - cluster 0': 0.6253369199550588,
'cluster 3 - cluster 1': 0.5018818999581539,
'cluster 3 - cluster 2': 0.10304450050414352,
'cluster 3 - cluster 3': 0.9916374307350072,
'cluster 3 - cluster 4': 0.7695534861413911,
'cluster 4 - cluster 0': 0.5972522450202913,
'cluster 4 - cluster 1': 0.4806080638937612,
'cluster 4 - cluster 2': 0.028939591901842553,
'cluster 4 - cluster 3': 0.7695534861413911,
'cluster 4 - cluster 4': 0.9711401898057233}
```

Na tym etapie wydawało się, że metoda analizy podobieństwa może zdać egzamin przy wyszukiwaniu powiązanych ze sobą dokumentów, jednak hipoteza ta została na dalszym etapie realizacji prac zweryfikowana negatywnie.

Aby przejść do dalszych prac konieczne było wykonanie szeregu działań mających na celu przygotowanie odpowiedniej infrastruktury i danych do dalszych eksperymentów, w tym:

1. Przygotowanie infrastruktury umożliwiającej wydajne przechowywanie i przetwarzanie (przeszukiwanie) wektorowej bazy dokumentów.

2. Rozwinięcie potoku przetwarzania dokumentów o wektoryzację i tworzenie wektorowej reprezentacji dokumentów.
3. Masowa wektoryzacja wszystkich zbiorów dokumentów.
4. Stworzenie narzędzi i infrastruktury umożliwiającej wielokrotne wykonywanie eksperymentów na zgromadzonej bazie dokumentów dla algorytmów z różnymi parametrami.

Jako miarę do weryfikacji jakości wyszukiwania powiązanych dokumentów przyjęto wartość procentową poprawnie zidentyfikowanych par dokumentów wobec całości testowanego zbioru. Zbiorem testowym dla tego eksperymentu było 2300 par dokumentów, które zostały ręcznie zidentyfikowane i powiązane ze sobą. W konsekwencji dla każdego z dokumentów wejściowych (faktury) wyszukiwano powiązaną umowę na zasadzie wyszukiwania dokumentów podobnych co do treści przy założeniu kosinusowej miary podobieństwa.

Dokument wejściowy pojawiający się w skrzynce wejściowej jest wstępnie przetworzony zgodnie z zaprojektowanym wcześniej procesem.

Treść dokumentu podlega sumaryzacji ekstraktywnej, to jest na podstawie statystyki wystąpienia słów kluczowych (po uprzednim usunięciu "stop words" i nieistotnych części mowy jak zaimki) wyznaczane są zdania, które zawierają najbardziej znaczące słowa kluczowe.

Podsumowanie podlega wektoryzacji, to jest poszczególne słowa są kodowane zgodnie z przyjętym modelem.

Jeśli do korpusu dokumentów ma zostać dodany nowy dokument to jest on zapisywany w bazie w postaci wektorowej.

Jeśli mają być pobrane z rejestru dokumenty powiązane to generowane jest zapytanie do bazy wektorowej, które zwraca top N dokumentów posortowanych według cosinusowej miary podobieństwa.

Na tym etapie prac okazało się, że **niezależnie od przyjętej metody wektoryzacji treści dokumentów poprawność identyfikowanych przez algorytm par dokumentów jest bardzo niska, na poziomie 10%.**

Bliższa analiza tego zjawiska wykazała, że przyczyną jest tutaj bardzo mały poziom podobieństwa tekstowego pomiędzy treścią faktury, a treścią powiązanej z nim umowy.

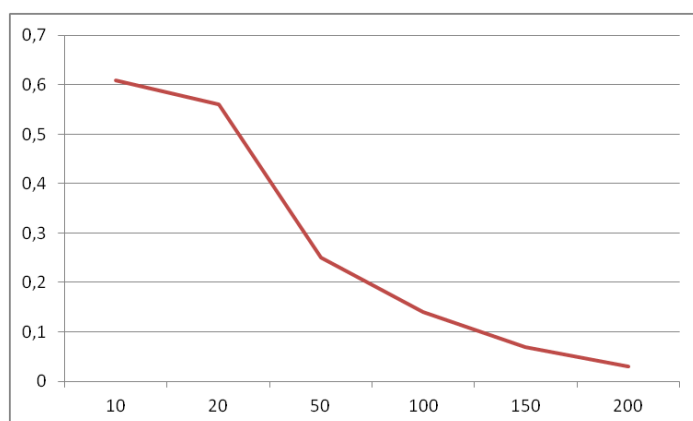
Zidentyfikowano 2 główne przyczyny takiego stanu:

1. W treści umowy występuje bardzo duży "szum tła" związany z dużą ilością słów języka prawniczego, które nie mają istotnego znaczenia w kontekście merytorycznej zgodności dokumentów. Statystyki tych słów przesuwają wektory reprezentujące umowy w taki sposób, że są one bliższe innym umowom i np. ogólnej korespondencji prawniczej bardziej niż powiązanym fakturom.

2. Opis przedmiotu na fakturze często nie zawiera tych samych słów, które występują w umowie. Zasadniczo znaczenie semantyczne przedmiotu na fakturze i przedmiotu umowy powinno być takie samo i modele, które biorą pod uwagę embedding słów powinny radzić sobie z tym problemem, ale jak wykazano już wcześniej na etapie klasyfikacji ze względu na błędy OCR oraz na specyfikę języka używanego w dokumentach występuje bardzo duży problem tzw. "out of vocabulary words" co dodatkowo biorąc pod uwagę duży szum tła dokumentu powoduje, że modele oparte na embedding słów nie radzą sobie z danymi używanymi w projekcie.

Biorąc pod uwagę powyższe obserwacje postawiona została dodatkowa hipoteza zakładająca, że lepsze wyniki wyszukiwania dokumentów na bazie podobieństwa może dać podział dokumentu na paragrafy i analiza podobieństwa mniejszych paragrafów zamiast całej treści dokumentu. Jeśli podobieństwo między paragrafami jest niskie to taki paragraf będzie odrzucany. Jeśli podobieństwo będzie wysokie to dany paragraf będzie brany pod uwagę i będzie podnosił prawdopodobieństwo powiązania danych dwóch dokumentów. W ten sposób jako powiązane dokumenty byłyby wybierane takie dokumenty, które mają najwięcej wspólnych paragrafów.

Aby zweryfikować powyższą hipotezę wprowadzono do procesu tzw. chunking (fragmentacja), czyli podział dokumentu na fragmenty (paragrafy). Kluczowym parametrem jest tutaj rozmiar fragmentu. Po wykonaniu kilku prób z różną wielkością fragmentów zdecydowano się przeprowadzić serię eksperymentów zakładającą badanie podobieństwa między dokumentami na bazie fragmentów różnej wielkości obejmującej od 200 do 10 słów. Na poniższym wykresie przedstawiono zależność między wielkością fragmentu, a skutecznością łączenia dokumentów w pary:



Jak widać okazało się, że **poprawność łączenia dokumentów w pary jest odwrotnie proporcjonalna do wielkości fragmentów. Dla dużych fragmentów obejmujących 200 słów i więcej, skuteczność jest dalece niesatysfakcjonująca. Dla mniejszych fragmentów udaje się uzyskać poprawność na poziomie 30-60%, co i tak nie jest wartością satysfakcjonującą w kontekście zastosowań realnych.**

Bliższa analiza danych wykazała, że de facto o połączeniu dokumentu w parę decyduje głównie paragraf zawierający dane stron umowy / transakcji. Dotyczy to 95% dokumentów, które zostały poprawnie połączone w pary. Taki wniosek sugeruje, że w samych danych istnieje silna korelacja dokumentów poprzez metadane i korelacja ta jest o wiele silniejsza niż korelacja przez ogólnie pojętą treść dokumentów. Bliższa analiza zjawiska potwierdza, że wybór danych kontrahenta jako głównego elementu definiującego korelację między dokumentami jest poprawna w około 70% przypadków. Pozostałe 30% przypadków obejmuje głównie sytuacje, gdzie istnieje więcej niż jeden dokument dotyczący danego kontrahenta. W takim przypadku okazuje się, że kolejnym elementem mającym silny wpływ na korelację dokumentów są numery umów i postępowań. Ostatnim elementem, który został zaobserwowany w analizie danych, że miał pozytywny wpływ na korelację dokumentów są fragmenty zawierające sam wyodrębniony opis przedmiotu umowy ale tylko w przypadkach, w których użyto do opisu na obu dokumentach tych samych słów.

Wyniku przeprowadzenia kilkudziesięciu eksperymentów obejmujących różne sposoby wektoryzacji oraz różny rozmiar fragmentów uzyskano wyniki, które pokazują, że wpływ rodzaju wektoryzacji nie jest istotny, a wpływ wielkości fragmentów bardzo duży, przy czym analiza zawartości fragmentów mających największy wpływ pokazuje, że de facto najsilniejszym czynnikiem korelacji między dokumentami są metadane, a nie treść dokumentów. Zgodność metadanych obejmujących dane kontrahentów, numery dokumentów i daty (w tej kolejności) jest warunkiem koniecznym, choć niewystarczającym, do stwierdzenia, że dwa dokumenty są powiązane. Analiza zawartości tekstowej ma sens tylko dla wydzielonych paragrafów zawierających opis przedmiotu umowy i może być użyta tylko jako warunek uzupełniający.

Z powyższych rozważań wynika jednoznacznie, że konieczne było opracowanie algorytmu korelowania dokumentów bazującego na tożsamości encji i przyjęcie tego algorytmu jako podstawowego algorytmu. Algorytm bazujący na podobieństwie tekstów może być wykorzystywany tylko do dodatkowej walidacji przedmiotu faktury w odniesieniu do przedmiotu umowy, co jest jednak nadal możliwe tylko przy spełnieniu szeregu założeń (prawidłowa ekstrakcja tych elementów z treści dokumentów i realne podobieństwo zapisów w obu dokumentach).

W dalszej kolejności opracowano więc algorytm wykorzystujący korelację dokumentów na podstawie tożsamości encji. Główne założenie algorytmu wyszukiwania na podstawie tożsamości encji zakłada, że dla danego dokumentu wejściowego wyszukujemy w rejestrze dokumentów takie dokumenty, w których występują te same wartości metadanych. Jest to rodzaj wyszukiwania Query By Example, gdzie wartości metadanych są wejściem do algorytmu generującego zapytanie do bazy danych.

Skuteczność poprawnego wykrywania par dokumentów na bazie tego algorytmu w zbiorze testowym wyniosła 78%.

Główne prace wykonywane na tym etapie polegały na:

1. Opracowaniu mechanizmu generowania zapytań do bazy danych poprzez tworzenie szablonów zapytań typu Query By Example uzupełnianych metadanymi odczytanymi z dokumentu wejściowego.
2. Pobieranie i scoring pobranych dokumentów w celu wybrania powiązanego dokumentu do walidacji krzyżowej.
3. Iteracyjne doskonalenie mechanizmów przetwarzania tekstu i ekstrakcji danych z uwzględnieniem wpływu na działanie mechanizmów korelacji dokumentów.

Bliższa analiza błędów związanych z korelacją dokumentów przez tożsamość encji pokazała, że cały czas kluczowym czynnikiem wpływającym na jakość działania systemu jest jakość ekstrakcji danych, która z kolei wynika z jakości tekstu dokumentów, co z kolei jest związane z jakością procesu OCR. Pokazuje to warstwową architekturę całego systemu, gdzie kolejne warstwy nie mogą działać lepiej, niż pozwalają na to warstwy niższe i **wymagane jest ciągłe, iteracyjne usprawnianie całego rozwiązania w miarę rozwoju kolejnych warstw.**

2. Opracowanie języka modelowania reguł biznesowych opartego na składni języka naturalnego.

Język musi umożliwiać tworzenie szablonów reguł w postaci wyrażen języka o składni zbliżonej do języka naturalnego tak, aby użytkownik mógł w łatwy sposób tworzyć szablony istotnych dla niego reguł biznesowych i aby było możliwe łatwe mapowanie reguł na wyrażenia języka naturalnego. W ramach zadania opracowana została składnia wyrażen języka oraz kompilator składni do postaci instrukcji działających na obiektach kontekstu biznesowego.

Celem tego zadania było opracowanie sposobu reprezentacji reguł odczytanych z treści dokumentów w taki sposób, aby docelowo mogły być one użyte do ewaluowania zależności między dokumentami i przeprowadzania w ten sposób tzw. walidacji krzyżowej. W związku z powyższym było to zadania oparte głównie o badanie istniejącej literatury tematu, w celu podjęcia decyzji o kierunkach dalszych prac takich jak wykorzystanie istniejących rozwiązań lub utworzenie własnego rozwiązania. W tym celu dokonano szerokiej analizy literatury obejmującej dwa główne kierunki badań:

1. Ekstrakcja z tekstu reguł do postaci strukturalnej (formalnej i wykonywalnej).
2. Translacja tekstu na języki strukturalne (formalne).

W konsekwencji przeprowadzonych analiz zaprojektowano gramatykę języka, która spełnia warunki konieczne do osiągnięcia zakładanych w projekcie cech systemu i kamieni milowych.

Wybrane najważniejsze pozycje literatury, które były przedmiotem rozważań:

1. Judging Amy: Automated Legal Assessment using OWL 2 Saskia van de Ven, Rinke Hoekstra, Joost Breuker, Lars Wortel, and Abdallah El-Ali 2008
2. Rules from Online Text Saeed Hassanpour, Martin J. O'Connor, Amar Das Stanford Center for Biomedical Informatics Research Stanford, CA, U.S.A. 2011

3. The Semantic of Business Vocabulary and Business Rules: An Automatic Generation From Textual Statements ABDELLATIF HAJ , ABDESSAMD JARRAR , YOUSSEF BALOUKI, AND TAOUFIQ GADIR 2021
4. Automatic Detection and Semantic Formalization of Business Rules Cheikh Kacfeh Emani¹ 2014
5. SBVR based Business Contract and Business Rule IDE Aqueo Kamada Guido Governatori , Shazia Sadiq , 2010
6. Transfer Learning for Deontic Rule Classification: Davide LIGA a,b,1, Monica PALMIRANI University of Luxembourg bAlma Human-AI, University of Bologna 2022
7. Deontic Reasoning for Legal Ontologies Cheikh Kacfeh Emani & Yannis Haralambous 2019
8. Agent-Specific Deontic Modality Detection in Legal Language Abhilasha Sancheti, Aparna Garimella, Balaji Vasani Srinivasan, Rachel Rudinger 2022
9. Cognitively Rich Framework to Automate Extraction And Representation of Legal Knowledge, Srishty Saha and Karuna P. Joshi
10. Automated Knowledge Extraction from the Federal Acquisition Regulations System (FARS) Srishty Saha, Karuna P. Joshi, Renee Frank, Michael Aebig, Jiayong Lin
11. Automated Directive Extraction from Policy Texts, Karl Branting, Jim Finegan, David Shin, Stacy Petersen, Alex Lyte, Carlos Balhana, Craig Pfeifer
12. Combining NLP Approaches for Rule Extraction from Legal Documents, Mauro Dragoni, Serena Villata, Williams Rizzi, Guido Governatori
13. On Rule Extraction from Regulations Adam WYNERa and Wim PETERS, 2011
14. Transforming Natural Language Query to SPARQL for Semantic Information Retrieval, S Mahaboob Hussain, 2016
15. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning, Victor Zhong, Caiming Xiong, Richard Socher, 2018
16. SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning, Xiaojun Xu, Chang Liu, Dawn Song, 2018
17. Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation, Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, Dongmei Zhang, 2019
18. Content Enhanced BERT-based Text-to-SQL Generation, Tong Guo, Huilin Gao, 2019
19. Exploring Sequence-to-Sequence Models for SPARQL Pattern Composition Anand Panchbhai, Tommaso Soru and Edgard Marx 1 , 2020

Analiza dotychczasowego dorobku naukowego:

W pracy "On Rule Extractions From Legal Regulations" autorzy rozważają możliwości ekstrakcji reguł z dokumentów normatywnych dostępnych w "natywnej" postaci cyfrowej w formacie XML. Materiałem wejściowym są dokumenty w języku angielskim. Celem ekstrakcji nie jest ewaluacja reguł, a raczej podsumowywanie dokumentów pod kątem eksponowania najważniejszych elementów treści. Wynikiem prac autorów jest opracowanie klasyfikatora, który wybiera z tekstu najważniejsze reguły i przypisuje je do jednej z kilku kategorii. Używane

metody sprowadzają się do parsowania struktury zdań za pomocą biblioteki Stanford NLP, próby uogólnienia struktury gramatycznej m.in. zdań poprzez ujednolicanie synonimów i klasyfikacje tak uzyskanych wektorów. Klasyfikacja odbywa się przez dopasowanie do wzorców, które zostały określone ręcznie na podstawie heurystyk. Warto wspomnieć, że w tej pracy autorzy wskazują, że kluczowe dla powodzenia zadania jest zidentyfikowanie tzw. "agentów i tematów" reprezentujących koncepty w zdaniach, a w następnej kolejności identyfikacja kluczowych czasowników wyrażających modalność deontyczna. Autorzy wskazują także, że dużym problemem jest używanie bibliotek, które są trenowane na ogólnie dostępnych tekstach, takich jak artykuły internetowe, do analizy tekstów prawnych o innej, bardziej skomplikowanej strukturze gramatycznej.

Podobne podejście prezentują autorzy pracy "Combining NLP Approaches for Rule Extraction from Legal Documents". Główna koncepcja przedstawiona przez autorów zakłada wyjście od z góry założonej lekkiej ontologii, która jest przygotowana ręcznie w postaci heurystyk, które opisują struktury gramatyczne typowe dla wyrażeń deontycznych. Te struktury są następnie uogólniane przez zastosowanie synonimów WordNet-u. Podstawową wykorzystywaną biblioteką jest tutaj również Stanford NLP.

W pracy "Agent-Specific Deontic Modality Detection in Legal Language" autorzy również podejmują temat ekstrakcji i klasyfikacji fragmentów tekstu jako reguł deontycznych. Autorzy traktują ten problem wprost jako problem klasyfikacji wieloklasowej. Jednak idą dalej w porównaniu do wcześniejszych prac i metody regułowe traktują tylko jako tzw. "benchmark bazowy" z którym porównują różnego rodzaju klasyfikatory wywodzone z modeli bazujących na transformerach (BERT). Zaproponowane przez autorów klasyfikatory rzeczywiście osiągają w pewnych przypadkach lepszą skuteczność niż metoda bazująca na dopasowaniu reguł. Jednak spektrum porównywanych algorytmów jest wąskie i nie wiadomo dlaczego nie zostały użyte inne rodzaje prostszych (mniej złożonych obliczeniowo) klasyfikatorów np. takie, jakie były wykorzystywane w projekcie beneficjenta.

Szereg dostępnych prac, jak np. "Automated Directive Extraction from Policy Texts", "Automated Knowledge Extraction from the Federal Acquisition Regulations System (FARS)", "Thesaurus Enhanced Extraction of Hohfeld's Relations from Spanish Labour Law", "Cognitively Rich Framework to Automate Extraction and Representation of Legal Knowledge" - wszystkie te prace proponują różne rozwiązania problemu ekstrakcji reguł deontycznych zasadniczo sprowadzając ten problem do problemu klasyfikacji. Różnią się one między sobą rodzajem proponowanych klasyfikatorów oraz mniej lub bardziej pracowitym definiowaniem bazowych ontologii i powiązanych z nimi heurystyk.

Praca "Cognitively Rich Framework to Automate Extraction and Representation of Legal Knowledge" różni się od pozostałych prac tym, że podejmuje próbę ewaluacji wyekstrahowanych reguł w kontekście pewnych zdarzeń (spraw, ang. cases). Autorzy tej pracy podobnie jak beneficjent przyjęli, że ewaluacja reguł musi się sprowadzać do podstawiania odpowiednich zmiennych w miejscu zidentyfikowanych obiektów (faktów) i wyliczania wartości logicznych wyrażeń. Rodzaje wyrażeń do ewaluacji są przyjęte na podstawie klasyfikacji modalności deontycznej (permisyjna, obligatoryjna itd.), a sama klasyfikacja również wykorzystuje dopasowanie słów kluczowych w strukturach zdań.

Praca “Automatic Detection and Semantic Formalisation of Business Rules” idzie dalej w kierunku nie tylko ekstrakcji, ale także ewaluacji reguł. Zakres dziedzinowy dotyczy przepisów budowlanych. Zasadniczo koncepcja autorów sprowadza się znów do sformułowania wyjściowej “ontologii” w postaci szablonów reguł w języku SPARQL i powiązanych z nimi list słów kluczowych. Dana reguła jest stosowana jeśli zostanie odnaleziony tekst o odpowiedniej strukturze gramatycznej w której występuje wymagane słowo kluczowe lub jego synonim. Następnie ekstrahowane są obiekty (fakty), które są podstawiane do reguły. Z kolei w pracy “Agent-Oriented Business Rules: Deontic Assignments” autorzy założyli, że ekstrahowane reguły biznesowe mogą być w sposób formalny wyrażone w języku SQL lub w celu ewaluacji mogą być konwertowane do instrukcji IF-THEN-ELSE, które są dostępne praktycznie w każdym języku programowania.

Praca “The Semantic of Business Vocabulary and Business Rules: An Automatic Generation From Textual Statements” jest pracą, która w najbardziej kompleksowy sposób podejmuje temat ekstrakcji i ewaluacji reguł biznesowych czerpiąc jednocześnie z wcześniejszych dokonań. Przyjęta metoda ekstrakcji zakłada dopasowanie struktury gramatycznej parsowanego tekstu do zestawu z góry przyjętych gramatycznych wzorców heurystycznych. Następnie w wybranych fragmentach tekstu identyfikowane są kluczowe obiekty i czasowniki i na tej podstawie generowane są wyrażenia w języku SVBR.

Jako uzupełnienie do zagadnienia samej ekstrakcji reguł przeprowadzono analizę szeregu prac poruszających zagadnienie transformacji zapytań w języku naturalnym na języki formalne (wykonywalne). W pracy “Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning” Victor Zhong, Caiming Xiong, Richard Socher, 2018, autorzy stosują podejście wykorzystujące reinforcement learning dla głębokich sieci neuronowych. Pomimo uproszczenia zagadnienia, które polega na translacji zapytań w postaci tekstu naturalnego na SQL odnoszony do danych tabelarycznych osiągnięto skuteczność na poziomie do 50%. Podobne wyniki skuteczności (do 60%) metod translacyjnych wykazano w pracy “Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation” Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, Dongmei Zhang, 2019, gdzie autorzy stosują model BERT do translacji przy wykorzystaniu pośredniej, uproszczonej gramatyki zapytań. Praca “Exploring Sequence-to-Sequence Models for SPARQL Pattern Composition” Anand Panchbhai 3, Tommaso Soru and Edgard Marx, 2020 jako wyjściowy język formalny przyjmuje SPARQL, a do translacji wykorzystuje architekturę Neural SPARQL Machines. Opisywane w artykule wyniki również nie przekraczają granicy 60%.

Wnioski z przeprowadzonej analizy

Z powyższej analizy istniejącego dorobku naukowego wynikają następujące wnioski, które potwierdzają zasadność i aspekt nowości przeprowadzonych prac projektowych:

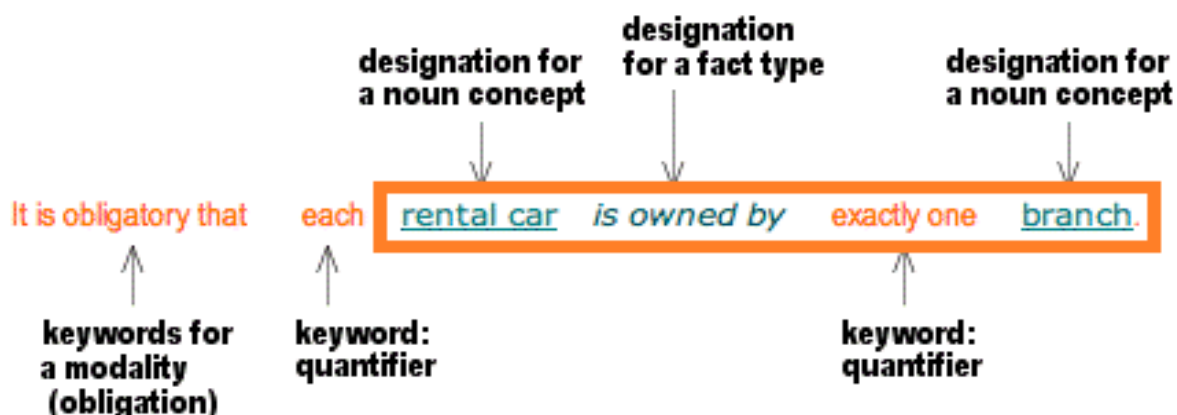
1. Istniejąca literatura przedmiotu dotyczy w zdecydowanej większości treści w językach innych niż język polski. Problemem z tym związanym jest duża złożoność gramatyki języka polskiego i brak dostępności odpowiednich danych treningowych.

2. Zakres dziedzinowy przeprowadzanych dotąd badań dotyczy głównie normatywnych tekstów prawnych, reguł inżynierii konstrukcji czy biomedycyny. Nie odnaleziono literatury odnoszącej się do dziedziny relacji B2B, co było głównym założeniem projektu. Z tego też powodu np. przyjęte w większości prac ogólne systemy klasyfikacji wyrażen (zakaz, zobowiązanie, zezwolenie) nie wpisują się w szczegółową dziedzinę projektu
3. Większość prac podejmuje temat samego aspektu ekstrakcji wyrażen deontycznych i ich klasyfikacji. Ewaluacja wyrażen jest poruszana zdecydowanie rzadziej, a jeśli jest to do ewaluacji są wykorzystywane języki i narzędzia formalne (OWL, SVBR, SPARQL etc.), które nie są przyjazne dla użytkowników biznesowych, cyt.: “One of the major drawbacks to using these methods is that the corpus for training is not always available, and businesspeople are not always familiar with (semi) formal models which limits their participation in the validation of generated models”.
4. Metody wykorzystujące translację języka naturalnego na dialekty formalne (SQL, SPARQL) nawet przy użyciu zaawansowanych modeli (transformery, głębokie sieci neuronowe) cechują się niską skutecznością na poziomie 50-60% poprawności pomimo wykorzystywania danych natywnych cyfrowo w języku angielskim.
5. **Biorąc pod uwagę punkt poprzedni oraz wnioski z poprzednich etapów pokazujące, że duże modele językowe i głębokie sieci neuronowe dają słabej jakości wyniki przy pracy na danych nie natywnych cyfrowo (poddanych procesowi OCR) ze względu na problem błędów w tekście (out of vocabulary words) słusznym jest wniosek, że wykorzystanie tych technik w projekcie jest nieuzasadnione, gdyż nie doprowadzi do osiągnięcia założonych wskaźników.**

Przyjęte rozwiązanie - język modelowania reguł

Przeprowadzona w ramach realizacji zadania analiza dostępnych rozwiązań doprowadziła do wyeliminowania z zakresu rozwiązań możliwych do zastosowania w projekcie rozwiązania wykorzystujące translację tekstu do języków formalnych takich jak SQL, SPARQL czy SBVR. Jednak analizując dostępne rozwiązania można wywieść wiele cech wspólnych powtarzających w cytowanych rozwiązaniach:

1. Modelowane reguły biznesowe mają zazwyczaj postać trójek $\{P1, R, P2\}$, gdzie P1 i P2 to koncepty reprezentujące pojęcia świata rzeczywistego połączone relacją R.
2. W języku naturalnym ta trójka występuje w postaci składowych zależności między częściami zdania definiującego daną regułę (źródło: <https://www.brcommunity.com/articles.php?id=b286>):



W wyniku analizy dziedziny obejmującej na cele testowe projektu przede wszystkim relacje gospodarcze opisane w umowach, których wynikiem realizacji jest wystawienie faktury, można zauważyć, że do wyrażenia głównych reguł biznesowych, których zadaniem jest walidacja krzyżowa faktury względem umowy wystarczy prosty język opisujący relacje między atrybutami poszczególnych obiektów. W takiej definicji problemu w trójce {P1, R, P2} podmiot P1 stanowi wartość atrybutu jednego z dokumentów, P2 drugiego, a R określa rodzaj relacji między nimi. Atrybuty te są ekstrahowane w wyniku działania wcześniej opisanych algorytmów ekstrakcji danych. Do wyrażania relacji między atrybutami opracowano prostą składnię język, którego przykład widoczny jest poniżej:

```
"Faktura/Wartość brutto" mniejszeRowne "Umowa/Wartość"
"Faktura/Numer konto" rowne "Umowa/Numer konta"
"Faktura/Data płatności" mniejszeRowne "Faktura/Data wystawienia" + "Umowa/Termin płatności"
```

Formalny opis składni również nie jest skomplikowany:

VALUE => "<NAZWA DOKUMENTU>/<NAZWA POLA METADANYCH>"

RULE_EXPR => VALUE_EXPR <NAZWA OPERATORA> VALUE_EXPR

VALUE_EXPR = VALUE | PYTHON_EXPR

Główne elementy składni opracowanego języka reguł to:

1. VALUE - Wyrażenie wartości atrybutu. Wyrażenie składa się z dwóch elementów: nazwy klasy dokumentu umieszczonego w kontekście i nazwy atrybutu tego dokumentu rozdzielonych znakiem "/".
2. RULE_EXPR - Wyrażenie reguły opisującej relacje między atrybutami. Relacja ta jest wyrażona przez nazwę operatora. Nazwa operatora mapowana jest na odpowiednią funkcję do której przekazywane są wartości wyrażeń (lewego i prawego argumentu operatora)
3. VALUE_EXPR - Wyrażenie, które jest wartością argumentu operatora. Każdy operator ma 2 argumenty: lewy i prawy (ArgL, ArgR) Wyrażenie to może być wartością atrybutu lub wyrażeniem języka Python. Przed ewaluacją wyrażenia w treści wyrażenia

rozwiązywane są najpierw do wartości zmiennych języka Python wszystkie odwołania typu VALUE.

Wyjściowo przewidziano implementację podstawowego zakresu operatorów:

1. równe EQ - Weryfikuje równość argumentów
2. mniejsze LT - sprawdza, czy argument lewy jest mniejszy niż prawy
3. większe GT - sprawdza, czy argument lewy jest większy niż prawy
4. mniejszeRówne LTEQ - sprawdza, czy argument lewy jest mniejszy lub równy prawemu
5. większeRówne GTEQ - sprawdza, czy argument lewy jest większy lub równy prawemu

Ta gramatyka jest wystarczająca do wyrażenia bezpośrednich relacji między dokumentami w kontekście. Jednocześnie może być ona dowolnie rozwijana przez wprowadzanie nowych operatorów oraz możliwość wywoływania funkcji języka Python w treści wyrażen.

Wynikiem tego zadania jest gramatyka języka typu DSL (Domain Specific Language) służąca do modelowania relacji krzyżowych pomiędzy atrybutami dokumentów występujących w danym kontekście transakcyjnym. Gramatyka ta została opracowana na bazie przeprowadzonych badań dostępnych rozwiązań, zamodelowana i zaimplementowana w docelowym rozwiązaniu.

3. Opracowanie algorytmu mapowania wyekstrahowanych z dokumentów wyrażen deontycznych na szablony reguł biznesowych.

Opracowany algorytm umożliwia prawidłowe mapowanie między szablonami reguł, a różnymi wyrażeniami występującymi w języku naturalnym, które mogą wyrażać to samo znaczenie semantyczne (tę samą regułę).

Zadanie to zostało zrealizowane poprzez opracowanie sposobu kodowania szablonów reguł jako wektorów cech, opracowanie sposobu reprezentacji wyekstrahowanych wyrażen jako wektorów cech oraz opracowanie algorytmu dopasowywania wyrażen do szablonów na podstawie reprezentacji wektorowych.

Założenia wstępne

Założeniem tego zadania było opracowanie algorytmu, który będzie mógł być wykorzystany do ekstrakcji reguł walidacji wartości atrybutów jednego dokumentu (np. Faktury) w kontekście wartości atrybutów drugiego, powiązanego dokumentu (np. Umowy) z uwzględnieniem zapisanej w umowie relacji między tymi atrybutami. Z analizy zebranych próbek wynikało, że najczęściej występujące zależności między atrybutami to relacja równości (kodowana jako EQ), relacja mniejsze lub równe (LTEQ) i większe lub równe (GTEQ). Opracowany algorytm na podstawie zapisu w umowie mówiącego np. na jakie konto ma być wykonana płatność wygeneruje regułę, która sprawdzi czy numer konta na fakturze jest równy temu podanemu w umowie.

Z przyjętej architektury rozwiązania wynikało, że główne komponenty rozwiązania zostały zidentyfikowane jako klasyfikatory, które prognozują na podstawie fragmentu tekstu:

1. Rodzaj operatora z puli zdefiniowanych w ontologii operatorów.
2. Nazwę pola argumentu lewego operatora z puli zdefiniowanych w ontologii pól.

3. Nazwę pola argumentu prawego operatora z puli zdefiniowanych w ontologii pól. Nazwa prawego argumentu operatora może być zastąpiona wartością literalnie określona w tekście. Wartość ta jest wyłuskiwana z tekstu przez ekstraktor wartości.

§ 3

1. Najemca będzie płacił Wynajmującemu na **konto** bankowe mBank **43 1140 2004 0000 3102 7830 3210** w miesiącach:

W przypadku wartości kwotowej umowy w przykładach pojawiały się warunki, które wprowadzają górne ograniczenie dla wartości.

Zamawiający zobowiązuje się zakupić a sprzedający sprzedać towary/usługi będące przedmiotem zamówienia.

Wartość zamówienia **nie przekroczy** kwoty brutto **1 000,00** zł, słownie: jeden tysiąc zł.

Zamawiający dokona zapłaty: gotówką/przelewem* w terminie 7 dni od otrzymania rachunku/faktury.

Z przedstawionych przykładów wynika, że jest możliwe przygotowanie klasyfikatora tekstów, który na podstawie analizy statystycznej słów występujących w tekście będzie prawidłowo prognozował jaki operator realizował będzie regułę zapisana w treści paragrafu (np. EQ, GTEQ, LTEQ). Zakładając przyjętą składnię języka reguł należy zauważyć, że oprócz predykcji samej klasy operatora konieczna jest także predykcja argumentów wejściowych (ArgL i ArgR). Argumenty te w trakcie wykonywania są wartościami pól odczytanych z dokumentów (np. kwota 1 000 pln), ale w zapisie ogólnym reguły muszą się odnosić do nazw porównywanych pól dokumentów (np. "Faktura/Wartość brutto" LTEQ "Umowa/Wartość brutto").

W trakcie realizacji zadania testowano trzy różne podejścia do problemu mapowania tekstu na szablony reguł, które będą zgodne z opracowanym modelem języka reguł:

1. Podejście bazujące na zadaniu ekstrakcji reguł jako zadaniu klasyfikacji. Przyjęto, że poszczególne składowe reguły będą wyznaczone przez wybrany rodzaj klasyfikatora tekstu.
2. Podejście bazujące na zadaniu ekstrakcji reguł jako zadaniu dopasowania wzorców. Przyjęto, że poszczególne składowe reguły będą wyznaczone przez dopasowanie wzorca gramatycznego lub tekstowego do treści zdania.
3. Podejście oparte na wykorzystaniu promptów i dużych modeli językowych. **Podejście to było testowane w końcowej fazie realizacji projektu w odpowiedzi na popularyzację i pojawienie się dostępności dużych modeli językowych na licencjach typu Open Source.** Przyjęto, że poszczególne składowe reguły będą wyznaczone jako odpowiedź na zapytanie kierowane do silnika LLM przy uwzględnieniu treści dokumentu jako danych promptu.

Ad. 1. Podejście bazujące na zadaniu ekstrakcji reguł jako zadaniu klasyfikacji.

Z tak postawionego zadania wynikała konieczność wytrenowania 3 rodzajów klasyfikatorów:

1. Klasyfikator operatora przewidujący na podstawie zawartości paragrafu o jakie porównywanie chodzi.
2. Klasyfikator, który będzie przewidywał jakie pole jest argumentem lewym operacji.
3. Klasyfikator, który będzie przewidywał jakie pole jest argumentem prawym operacji.

Na podstawie przygotowanego zbioru danych można zauważyć, że takie wnioskowanie jest możliwe na podstawie treści odpowiednich paragrafów:

- 2) Za wykonanie przedmiotu umowy Strony ustalają wynagrodzenie w wysokości 4.797,00 zł brutto (słownie: cztery tysiące siedemset dziewięćdziesiąt siedem złotych brutto, 00/100).
- 3) Termin wykonania umowy ustala się od dnia jej podpisania do dnia 19.05.2023r.

§ 6

1. Strony uzgadniają wynagrodzenie za wykonanie całości przedmiotu umowy na kwotę brutto 34 440,00 zł (słownie: trzydzieści cztery tysiące czterysta czterdzieści złotych brutto, 00/100 gr), w tym:

Zamawiający zobowiązuje się zakupić a Wykonawca sprzedać towary/ usługi* będące przedmiotem zamówienia.

Wartość zamówienia nie przekroczy kwoty brutto 650 zł / słownie: sześćset pięćdziesiąt złotych 00/100

Zamawiający dokona zapłaty gotówką / przelewem* w terminie 14.dni od otrzymania rachunku/faktury*.

§ 2

Ustala się następujący termin zakończenia wykonania całości prac 45 dni od podpisania umowy.

§ 3

1. Za wykonanie przedmiotu niniejszej umowy ustala się wynagrodzenie ryczałtowe brutto w wysokości: 2.300 zł (słownie: dwa tysiące trzysta złotych).
1. Za wykonanie prac projektowych C1 - (w tym uzyskanie wymaganych decyzji i uzgodnień wraz z decyzją o pozwoleniu na budowę) Wykonawca otrzyma wynagrodzenie ryczałtowe w wysokości 16 000,00 zł netto + 23% VAT (słownie: szesnaście tysięcy zł zero gr) tj. 19 680,00 zł brutto (słownie: dziewiętnaście tysięcy sześćset osiemdziesiąt zł zero gr), zgodnie z ceną ofertową Wykonawcy.

§ 4

1. Termin wykonania przedmiotu umowy: od dnia podpisania umowy do dnia 30 czerwca 2023 r.

§ 5

1. Strony ustalają wynagrodzenie ryczałtowe za wykonanie przedmiotu umowy wraz z przeniesieniem majątkowych praw autorskich w kwocie 31 980,00 zł (wraz z podatkiem VAT) (słownie: trzydzieści jeden tysięcy dziewięćset osiemdziesiąt złotych 00/100).

§ 8

1. Termin wykonania przedmiotu umowy do 60 dni kalendarzowych od daty zawarcia umowy.

§ 9

1. Wynagrodzenie Wykonawcy za wykonanie przedmiotu umowy ustala się w formie ryczałtu na kwotę brutto 110 000,00 zł (słownie złotych: sto dziesięć tysięcy 00/100), łącznie z podatkiem VAT, zgodnie z ofertą Wykonawcy stanowiącą załącznik nr 1 do umowy.

Konieczne okazało się opracowanie odpowiedniego zbioru uczącego. Ze względu na dużą ilość pracy ręcznej opracowano algorytm, który pomagał w automatyzacji procesu przygotowywania danych treningowych:

1. Na podstawie algorytmu ekstrakcji (opracowanego wcześniej) jest możliwe określenie gdzie w tekście występuje wartość atrybutu.
2. Wychodząc od pozycji w tekście odnalezionego atrybutu wyznaczono zakres otoczenia tekstowego, które jest traktowane jako paragraf definiujący sposób działania operatora.

Na tej podstawie przygotowano zbiór treningowy umożliwiający wytrenowanie potrzebnych klasyfikatorów.

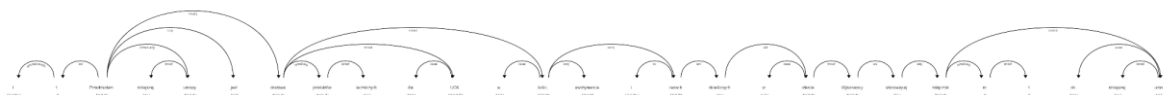
	A	B	C	D
1	tekst	operator	argL	argR
2	2) Za wykonanie przedmiotu umowy Strony ustalają wynagrodzenie w wysokości 4.797,00 zł brutto (słownie: cztery tysiące siedemset dziewięćdziesiąt siedem złotych brutto, 00/100).	EQ	Faktura/Wartość brutto	Umowa/Wartość brutto
3	3) Termin wykonania umowy ustala się od dnia jej podpisania do dnia 19.05.2023r.	LTEQ	Faktura/Data sprzedaży	Umowa/Termin
4	§s 1. Strony uzgadniają wynagrodzenie za wykonanie całości przedmiotu umowy na kwotę brutto 34 440,00 zł (słownie: trzydzieści cztery tysiące czterysta czterdzieści złotych brutto, 00/100 gr), w tym:	EQ	Faktura/Wartość brutto	Umowa/Wartość brutto
5	Zamawiający zobowiązuje się zakupić a sprzedający sprzedać towary/usługi będące przedmiotem zamówienia. Wartość zamówienia nie przekroczy kwoty brutto 1 000,00 zł, słownie: jeden tysiąc zł.	LTEQ	Faktura/Wartość brutto	Umowa/Wartość brutto
6	§3 1. Najemca będzie płacił Wynajmującemu na konto bankowe mBank 43 1140 2004 0000 3102 7830 3210 w miesiącach:	EQ	Faktura/Konto	Umowa/Konto
7	Zamawiający dokona zapłaty: ge-tówk,ą/przelewem* w terminie 7 dni od otrzymania rachunku/faktury.	LTEQ	Faktura/Termin płatności	Faktura/Data sprzedaży
8	Wartość zamówienia nie przekroczy kwoty brutto 650 zł / słownie: sześćset pięćdziesiąt złotych 00/100	LTEQ	Faktura/Wartość brutto	Umowa/Wartość brutto
9	Zamawiający dokona zapłaty gotówką / przelewem* w terminie 14.dni od otrzymania rachunku/faktury*.	GTEQ	Faktura/Termin płatności	Faktura/Data sprzedaży
10	2 Ustala się następujący termin zakończenia wykonania całości prac: 45 dni od podpisania umowy. 3	GTEQ	Faktura/Data sprzedaży	Umowa/Termin
	1. Za wykonanie przedmiotu niniejszej umowy ustala się wynagrodzenie	EQ	Faktura/Wartość brutto	Umowa/Wartość brutto

Na podstawie przygotowanego zbioru danych przeprowadzono trening i testy różnych klasyfikatorów (Logistic Regression, SGD, Support Vector) i różnych sposobów wektoryzacji tekstu (BOW, TFIDF, HASH). Niestety okazało się, że uzyskane wyniki są poniżej oczekiwań i uzyskane wartości średniej harmonicznej F1 klasyfikatorów są na poziomie 0,4-0,6. Po bliższej analizie okazało się, że przyczyną jest sposób wektoryzacji tekstu, który pomijał słowa nieistotne (tzw. stop words) takie jak "do" i "w". Tymczasem to właśnie subtelne różnice w wyrażeniach "wynagrodzenie w kwocie" lub "wynagrodzenie do kwoty" decyduje o rodzaju operatora (EQ lub LTEQ). Ponadto istotne są też pary słów, a nie samo wystąpienie lub nie poszczególnych słów kluczowych w tekście. Dlatego też wprowadzono odpowiednie zmiany w przetwarzaniu testu i zastosowano bi-gramy w wektoryzacji tekstu. To podejście podniosło nieznacznie wyniki treningu klasyfikatorów do wartości 0,5-0,7. W tej sytuacji podjęto decyzję o przetestowaniu alternatywnego podejścia heurystycznego do tworzenia reguł. Podejście to zakłada możliwość określania ręcznego reguł, które są oparte na wskazywaniu słów, które muszą wystąpić nie dalej niż N słów przed i słów, które muszą wystąpić nie dalej niż N słów po wystąpieniu w tekście wartości atrybutu, aby skutkować zastosowaniem danego rodzaju operatora. Przy zastosowaniu tego podejścia nie uzyskano istotnie lepszych wyników.

Ad. 2. Podejście bazujące na zadaniu ekstrakcji reguł jako zadaniu dopasowania wzorców.

W tym podejściu testowano możliwość stosowania dopasowania wzorców reprezentujących znane z góry reguły występujące w dziedzinie problemu do ich wystąpień w tekście. Podejście to nie bazuje oczywiście na prostym dopasowaniu wzorców tekstowych reprezentujących poszczególne reguły, ale na opracowanych wzorcach uogólnionych. Wykorzystywana jest tutaj analiza gramatyczna zdań i tzw. dependency parsing. Parser gramatyczny tworzy drzewo gramatyczne, w którym słowa powiązane są grupy zależności w zależności od ich funkcji w zdaniu. Tekst podlega również tokenizacji obejmującej doprowadzenie poszczególnych słów do formy bazowej. W ten sposób możliwe jest tworzenie dla każdej reguły uogólnionych

wzorców, które bazują na występowaniu typowych słów kluczowych w odpowiednich relacjach z innymi słowami w strukturze zdania. W toku prac opracowano kilkanaście wzorców reprezentujących typowe reguły istotne z punktu widzenia rozważanej dziedziny. Testy wykazały dużą skuteczność ekstrakcji (F1 rzędu 0,8) reguł przy użyciu dopasowania wzorców opartego na dependency parsingu:



DOBRY PRZYKŁAD DEPENDENCY PARSINGU - ZA JEGO POMOCĄ MOŻNA WYCIĄGNĄĆ TOKENY NUMERYCZNE, KTÓRE SĄ DZIEĆMI DLA TOKENA 'NIP'

```
nip_lst = []
nip = ""
for sent in doc.sents:
    for token in sent:
        if token.text.lower() == "nip":
            potential_nip_lst = [child for child in token.children]
            for ele in potential_nip_lst:
                if ele.text.isdigit():
                    nip += ele.text
                    if len(nip) == 10 and is_nip(nip) and nip == uck_nip:
                        nip = ""
                        continue
                    elif len(nip) == 10 and is_nip(nip) and nip != uck_nip:
                        nip_lst.append(nip)
                        nip = ""
                        continue

# pattern matching approach

nip_patterns = [
    [{'LOWER': 'nip'}, {'IS_PUNCT': True}, {'POS': 'NUM', 'OP': '+'}, {'SPACY': True}, {'POS': 'NUM', 'OP': '!'}],
    [{'LOWER': 'nip'}, {'POS': 'NUM', 'OP': '+'}, {'SPACY': True}, {'POS': 'NUM', 'OP': '!'}]
]

matcher = Matcher(nlp.vocab)
matcher.add('nip', nip_patterns)

if len(nip_lst) == 0:
    for match_id, start, end in matcher(doc):
        span = doc[start:end]
        nip = span.text
        nip_lst.append(nip)
```

```
def extract_agreement_nums(doc):
    agreement_nums = []

    for sent in doc.sents:
        for token in sent.text.split():
            if (any(yr == token.split('/')[0] for yr in list_of_years) or any(yr == token.split('.')[0] for yr in list_of_years)
                or any(yr == token.split('_')[0] for yr in list_of_years)) and not is_date(token.replace(',', '').replace('(', '').replace(')', '')):
                if token.startswith("/") or token.startswith("-") or token.startswith("_"):
                    continue
            agreement_nums.append(token)

    result = list(set(agreement_nums))
    return sorted(result, key=len)
```

FUNKCJA ZWRACAJĄCA ZDANIA ZAWIERAJĄCE ODPOWIEDNIE WZORCE DLA DAT

```
def get_date_sents(doc):
    date_sents = []
    for sent in doc.sents:
        rst = re.findall("[0-9]{1,2}\.[0-9]{1,2}\.[0-9]{4}", sent.text.replace(" ", ""))
        if len(rst) > 0:
            date_sents += (sent.text, rst),
        else:
            rst = re.findall("[0-9]{1,2}\\[0-9]{1,2}\\[0-9]{4}", sent.text.replace(" ", ""))
            if len(rst) > 0:
                date_sents += (sent.text, rst),
    return date_sents
```

FUNKCJA ZWRACAJĄCA KONKRETNE ZDANIE Z DATĄ PODPISANIA/ZAWARCIA UMOWY ZA POMOCĄ WYSZUKIWANIA FRAZ KLUCZOWYCH

```
def extract_agreement_conclusion(doc):
    phrase_matcher = PhraseMatcher(nlp.vocab)
    agreement_phrases = ["umowa zostaje zawarta", "umowa została zawarta", "w dniu", "pomiędzy"]
    agreement_patterns = [nlp(text) for text in agreement_phrases]
    phrase_matcher.add('agreement', None, *agreement_patterns)

    agreement_conclusion = None
    for sent, dt in get_date_sents(doc):
        for match_id, start, end in phrase_matcher(nlp(sent)):
            if nlp.vocab.strings[match_id] in ["agreement"]:
                agreement_conclusion = sent, dt
    return agreement_conclusion
```

FUNKCJA ZWRACAJĄCA OKRES OBOWIĄZYWANIA UMOWY (Z TEKSTU)

```
def extract_period_of_validity(doc):
    period_of_validity_patterns = [
        [{'LEMMA': 'miesiąc'}, {'LOWER': 'od'}, {'LEMMA': 'data'}, {'LOWER': 'zawarcia'}],
        [{'LOWER': 'okres'}, {'LOWER': 'obowiązywania'}],
        [{'LOWER': 'w'}, {'LOWER': 'okresie'}, {'POS': 'NUM'}, {'LEMMA': 'miesiąc'}],
        [{'LOWER': 'w'}, {'LOWER': 'terminie'}, {'POS': 'NUM'}, {'POS': 'NOUN', 'DEP': 'nmod:arg'}, {'LOWER': 'od'},
        {'LOWER': 'daty'}, {'LOWER': 'podpisania'}, {'LOWER': 'umowy'}]
    ]

    matcher = Matcher(nlp.vocab)
    matcher.add('validity', period_of_validity_patterns)

    period_of_validity = None
    for sent in doc.sents:
        for match_id, start, end in matcher(sent):
            if nlp.vocab.strings[match_id] in ["validity"]:
                period_of_validity = sent.text
    return period_of_validity
```

FUNKCJA POSZUKUJĄCA ZDAŃ Z OPISEM PRZEDMIOTU UMOWY NA PODSTAWIE ODPOWIEDNIICH WZORCÓW GRAMATYCZNYCH (NIE MYLIĆ Z REGEXAMI!)

```
def extract_deal_subject_start(doc):
    deal_subject_start_patterns = [
        [{'POS': 'ADP'}, {'POS': 'NOUN'}, {'POS': 'ADJ'}, {'POS': 'NOUN'}, {'LOWER': 'postępowania'}], # w wyniku przeprowadzonego postępowania
        [{'LOWER': 'przedmiotem'}, {'POS': 'ADJ'}, {'POS': 'NOUN'}, {'POS': 'AUX'}], # przedmiotem niniejszej umowy jest
        [{'LEMMA': 'umowa'}, {'LOWER': 'o'}, {'LOWER': 'następującej'}, {'LOWER': 'treści'}], # umowę o następującej treści
        [{'LOWER': 'wykonawca'}, {'POS': 'NOUN'}, {'LOWER': 'wydzierżawia'}, {'POS': 'VERB'}],
        [{'LOWER': 'zamawiającemu'}, {'POS': 'ADJ'}, {'POS': 'ADP'}, {'POS': 'NOUN'}, {'OP': '{3}'}], # wykonawca wydzierżawia zamawiającemu na okres tr
        [{'LOWER': 'wynajmujący'}, {'POS': 'NOUN'}, {'POS': 'VERB'}, {'LOWER': 'najemcy'}]]

    matcher = Matcher(nlp.vocab)
    matcher.add("deal_subject_start", deal_subject_start_patterns)

    deal_subject_start = None

    matches = matcher(doc)
    for match in matches:
        pattern_id = nlp.vocab.strings[match[0]]
        span = doc[match[1]:match[-1]]
        txt = span.text
        txt_start = span.start
        txt_end = span.end
        deal_subject_start = txt, txt_start, txt_end

    return deal_subject_start
```

FUNKCJA WYSZUKUJĄCA ZDANIA Z WARTOŚCIĄ NETTO/BRUTTO BAZUJE GŁÓWNIENIE NA DEPENDENCY PARSINGU PO PRZEPROCESOWANIU DOKUMENTU PRZEZ MODEL

```
def extract_agreement_value(doc):
    price_sent_patterns = [
        [{'LEMMA': 'wartość'}, {'LEMMA': 'netto'}],
        [{'LEMMA': 'wartość'}, {'LEMMA': 'brutto'}],
        [{'LOWER': 'cena'}, {'LOWER': 'całej'}, {'LOWER': 'oferty'}],
        [{'LOWER': 'szacunkowa'}, {'LOWER': 'wartość'}, {'POS': 'ADJ'}, {'POS': 'NOUN'}],
        [{'LOWER': 'zamawiający'}, {'LOWER': 'zapłaci'}, {'LOWER': 'wykonawcy'}],
        [{'LOWER': 'wartość'}, {'LOWER': 'niniejszej'}, {'LOWER': 'umowy'}]]

    matcher = Matcher(nlp.vocab)
    matcher.add("price_sent", price_sent_patterns)
    result_sents = []

    matches = matcher(doc)
    for match_id, start, end in matches:
        pattern_id = nlp.vocab.strings[match_id]
        span = doc[start:end]
        result_sents.append((pattern_id, span.sent))

    # print(list(set(result_sents)))

    price_patterns = [
        [{'LOWER': 'netto'}, {'IS_PUNCT': True}, {'DEP': 'nummod:gov', 'OP': '+'}, {'LOWER': 'zł'}],
        [{'LOWER': 'netto'}, {'IS_PUNCT': True}, {'DEP': 'nummod:gov', 'OP': '+'}, {'LOWER': 'pln'}],
        [{'LOWER': 'netto'}, {'DEP': 'nummod:gov', 'OP': '+'}, {'LOWER': 'zł'}],
        [{'LOWER': 'netto'}, {'DEP': 'nummod:gov', 'OP': '+'}, {'LOWER': 'pln'}],
        [{'LOWER': 'brutto'}, {'IS_PUNCT': True}, {'DEP': 'nummod:gov', 'OP': '+'}, {'LOWER': 'zł'}],
        [{'LOWER': 'brutto'}, {'IS_PUNCT': True}, {'DEP': 'nummod:gov', 'OP': '+'}, {'LOWER': 'pln'}],
        [{'LOWER': 'brutto'}, {'DEP': 'nummod:gov', 'OP': '+'}, {'LOWER': 'zł'}],
        [{'LOWER': 'brutto'}, {'DEP': 'nummod:gov', 'OP': '+'}, {'LOWER': 'pln'}],
        [{'LEMMA': 'wynosić'}, {'POS': 'NUM', 'OP': '+'}, {'TEXT': '('}]]

    matcher = Matcher(nlp.vocab)
    matcher.add("price", price_patterns)
```

Ad. 3. Podejście oparte na wykorzystaniu promptów i dużych modeli językowych

W tym podejściu testowano możliwość wykorzystania dużych modeli językowych (LLM) do ekstrakcji informacji o wartościach i ograniczeniach wyrażanych w treści umowy. Podejście zakładało opracowanie tzw. promptów o odpowiedniej strukturze do ekstrakcji wymaganej wiedzy z tekstu. Struktura promptu zakłada podanie w kontekście konwersacji treści umowy jako wiedzy kontekstowej (in context learning). Na bazie tej wiedzy w pamięci roboczej modelu

LLM inicjowany jest kontekst z którego po zadaniu pytania model powinien wnioskować odpowiedź. Podejście to było testowane pod koniec prac projektowych po pojawieniu się dostępnych publicznie modeli LLM dla języka polskiego. Podejście to dawało wstępnie obiecujące efekty, jednak zauważono szereg problemów, które stwarzały ryzyko tego, że ostateczny produkt będzie mało użyteczny. Po pierwsze czas wnioskowania modelu LLM był bardzo długi w stosunku do innych rozwiązań. Testowano kilka modeli LLM i uzyskiwano zdecydowanie gorsze wyniki dla mniejszych modeli, z kolei modele większe sprawiały problemy z uruchomieniem i stosowanie ze względu na duże zapotrzebowanie na zasoby sprzętowe długi czas odpowiedzi. Kolejnym i dużo większym problemem była skłonność modeli LLM do tzw. halucynacji czyli generowania odpowiedzi, które nie występują w przeszukiwanym tekście. Nie ma żadnej kontroli, czy odpowiedź jest dawana przez model na podstawie tekstu dostarczonego w kontekście promptu, czy jest wygenerowana przez model na podstawie zapamiętanej w modelu wiedzy ogólnej wytworzonej w procesie trenowania. O ile w pozostałych modelach jest możliwe wskazanie np. fragmentu tekstu, który został w jakiś sposób zakwalifikowany jako odpowiedź to w modelach LLM taka inżynieria odwrotna jest bardzo trudna. Ze wskazanych względów podejście bazujące na dużych modelach językowych uznano za obiecującą alternatywę, ale wymagającą dalszego rozwoju technik związanych z weryfikacją rzetelności odpowiedzi modelu.

DOKONAŁIŚMY WYBORU MODELU 'POLISH-QA-V2' TRENOWANEGO DATASECIE 'POQUAD'

Przykład poprawnych odpowiedzi modelu dla zapytań o wartość i termin płatności:

```
question = "Jaka jest wartość netto umowy?"

start = time()

print(
    question_answerer(
        question=question,
        context=context.replace("\n", " ")
    )
)

end = time()

print("-----")
print(f"Ground truth: {df['netValue'][row_no]}")
print("-----")
print(f"Czas odpowiedzi: {end - start}")

{'score': 0.7846311926841736, 'start': 1836, 'end': 1853, 'answer': ' 1 073 937,10 PLN'}
-----
Ground truth: 1073937.1
-----
Czas odpowiedzi: 19.052616119384766
```

```
question = "Jaki jest termin płatności po wystawieniu faktury wskazany w umowie?"

start = time()

print(
    question_answerer(
        question=question,
        context=context.replace("\n", " ")
    )
)

end = time()

print("-----")
print(f"Ground truth: {df['payment'][row_no]}")
print("-----")
print(f"Czas odpowiedzi: {end - start}")

{'score': 0.9689794778823853, 'start': 6460, 'end': 6467, 'answer': ' 60 dni'}
-----
Ground truth: 60 DNI OD OTRZYMANIA FAKTURY
-----
Czas odpowiedzi: 19.611428260803223
```

Przykład błędnej odpowiedzi na bazie halucynacji modelu.

```
question = "Jaka jest data zakończenia niniejszej umowy?"

start = time()

print(
    question_answerer(
        question=question,
        context=context.replace("\n", " ")
    )
)

end = time()

print("-----")
print(f"Ground truth: {df['endDate'][row_no]}")
print("-----")
print(f"Czas odpowiedzi: {end - start}")

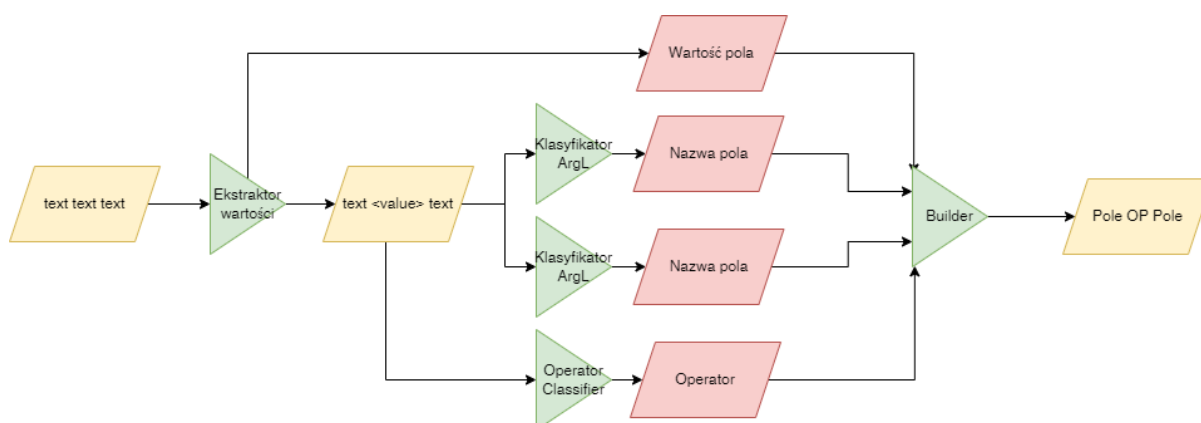
{'score': 0.9262522459030151, 'start': 10111, 'end': 10123, 'answer': ' 4 miesiące.'}
-----
Ground truth: 02-07-2020
-----
Czas odpowiedzi: 19.413421392440796
```

UZYSKANE WYNIKI I WNIOSKI

W wyniku przeprowadzonych prac zaprojektowano ostatecznie architekturę algorytmu, która umożliwia detekcję i generowanie odpowiednich operatorów dla tekstu paragrafu dokumentu.

Sposób działania algorytmu jest następujący:

1. Tekst jest skanowany przez ekstraktor wartości atrybutów. W przypadku znalezienia atrybutu tekst jest przekazywany dalej w celu określenia właściwego operatora.
2. Klasyfikator statystyczny lub regułowy określa rodzaj operatora do zastosowania.
3. KlasyfikatorArgL określa jakie pole jest lewym argumentem wejściowym klasyfikatora.
4. KlasyfikatorArgR określa jakie pole jest prawym argumentem wejściowym klasyfikatora.
5. Następnie składana jest odpowiednia reguła. Wygenerowane w ten sposób reguły mogą być użyte w kroku procesu odpowiedzialnym za walidację krzyżową dokumentów w procesie.



W wyniku przeprowadzonych badań dokonano testu skuteczności wyznaczania reguł dla wybranych kluczowych atrybutów dokumentów przy użyciu różnych strategii generowania reguł. Testy były wykonywane na próbce 200 dokumentów reprezentujących umowy ze zbioru testowego. Reguły są wyznaczane na podstawie umów ponieważ to właśnie w treści umowy są one zawarte.

Wyniki testów przedstawiono poniżej:

	kwota netto	termin płatności	kontrahent	konto	średnia
dopasowanie reguł	0,74	0,69	0,93	0,94	0,83

klasyfikacja	0,71	0,65	0,79	0,85	0,75
prompt engineering	0,53	0,52	0,76	0,82	0,66

Jak wynika z przeprowadzonych eksperymentów podejście regułowe daje największą skuteczność ekstrakcji reguł w badanym zbiorze testowym. Niestety okupione jest to dużą ilością czasu poświęconą na tworzenie heurystycznych reguł odzwierciedlających różne wzorce gramatyczne reprezentujące poszczególne elementy każdej reguły. Z kolei podejście oparte na trenowaniu klasyfikatorów z danych wykorzystuje uogólnione wnioskowanie oparte na wektoryzacji danych, ale z kolei wymaga pracy nad przygotowaniem odpowiedniej ilości i jakości danych treningowych. Obiecujące wyniki (relatywnie do czasu poświęconego tej technice) daje metoda prompt engineeringu z wykorzystaniem modeli LLM. Tutaj problemem jest jednak bardzo duży czas odpowiedzi modelu na zapytania oraz skłonności do halucynacji (dawania odpowiedzi nie bazujących na dostarczonych danych). W tej metodzie dostosowanie do konkretnej dziedziny wymaga zmian w promptach.

Cechy wszystkich podejść mają duże znaczenie w przypadku dostosowywania systemu do nowych dziedzin. W przypadku braku dużej ilości danych możliwe jest zastosowanie podejścia regułowego, a brak danych treningowych jest częstym problemem w przypadku dostosowywania modeli językowych do niszowych dziedzin.

4. Implementacja ewaluatora reguł w kontekście zbioru powiązanych dokumentów.

W ramach niniejszego zadania została stworzona architektura silnika wykonawczego pozwalającego na załadowanie powiązanych dokumentów do kontekstu biznesowego, załadowanie reguł do kontekstu biznesowego oraz uruchomienie reguł i zwrócenie wyniku.

Niniejsze zadanie miało w dużej mierze charakter implementacyjny polegający na realizacji praktycznej wcześniej opracowanych koncepcji i ich walidacja w testach. Zadanie obejmowało następujące podzadania:

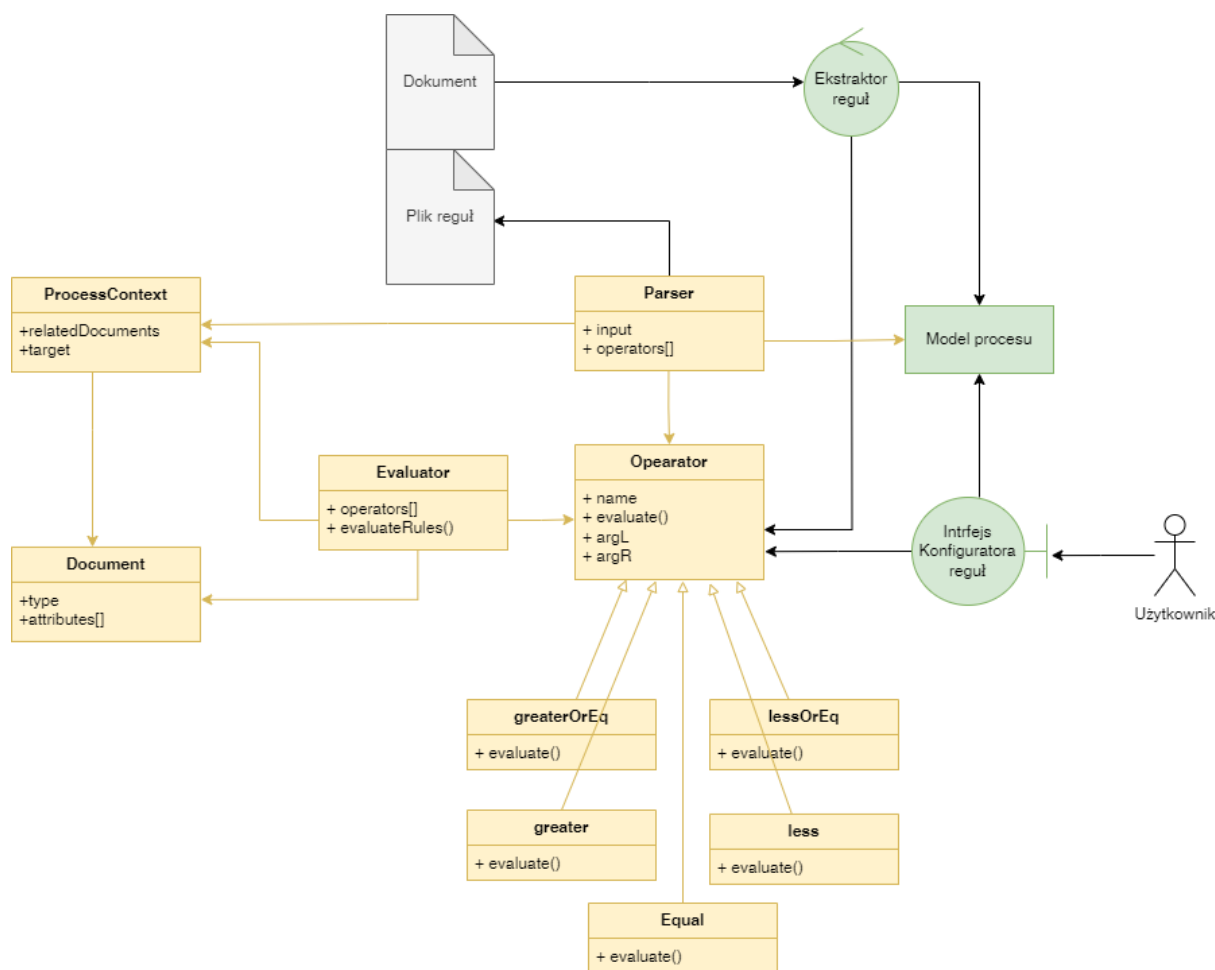
1. Implementacja modelu obiektowego reprezentującego elementy gramatyki języka.
2. Implementacja ewaluatora uwzględniająca wiązanie zmiennych (data binding) w kontekście.

Sercem rozwiązania jest architektura obiektowa składająca się z następujących elementów:

1. Document - obiekt przechowujący dane dokumentu w postaci listy atrybutów. Dla każdego rodzaju dokumentu (faktura, umowa, korespondencja) skonfigurowany może być inny zestaw metadanych. Z dokumentem powiązany jest również załącznik (plik) wejściowy oraz pośrednie artefakty przetwarzania (wynik OCR w postaci hOCR, tekst płaski dokumenty, obiekt JSON zawierający metadane).
2. ProcessContext - obiekt przechowujący dane kontekstowe procesu. Dane są przechowywane w postaci mapy zmiennych nazwa-wartość. Kluczowym elementem

kontekstu procesu jest inicjujący dokument wejściowy (zamienna target) i inne powiązane dane, w tym inne dokumenty pobrane z repozytorium do kontekstu przez algorytm wyszukiwania powiązanych dokumentów.

3. Evaluator - obiekt wykonujący ewaluację wyrażeń. Obiekt ten uruchamia właściwe operatory i wylicza właściwą wartość wyrażeń będących argumentami operatorów.
4. Operator - Obiekt realizujący logikę porównania wartości argumentów. Zwraca wartość logiczną true/false.
5. Parser - parser wyrażeń reguł biznesowych, który na podstawie tekstu reguł tworzy odpowiednią strukturę operatorów.
6. Ekstraktor reguł - proces treningowy, który na podstawie próbki dokumentów ma możliwość ekstrakcji reguł biznesowych zawartych w treści dokumentów zgodnie z opracowaną w poprzednim zadaniu metodyką.
7. Model procesu - przebieg procesu zamodelowany jako ciąg kolejnych czynności (zadań). Procesy są modelowane w formacie XML. Poszczególnym zdaniom odpowiadają odpowiednie elementy XMLa, które są następnie interpretowane przez silnik runtime w czasie wykonywania. Wykorzystywany jest silnik procesów bazujący na standardzie BPMN.
8. Konfigurator reguł - graficzny interfejs umożliwiający weryfikację i konfigurację reguł przetwarzania procesów.



W trybie roboczego wykonania proces przetwarzania wygląda następująco:

1. Serwis INBOX wybiera nowy plik i inicjuje wstępny proces przetwarzania.
2. Plik przechodzi przez skonfigurowany potok przetwarzania obejmujący ekstrakcję tekstu i inne wymagane kroki zgodnie z opracowaną definicją procesu.
3. Treść pliku podlega klasyfikacji i określany jest biznesowy typ dokumentu.
4. Na podstawie określonej klasy dokumentu uruchamiany jest odpowiedni proces ekstrakcji danych.
5. Do kontekstu procesu dodawany jest obiekt "target" reprezentujący wejściowy dokument.
6. Do kontekstu dodawane są dokumenty powiązane pobrane z repozytorium dokumentów na podstawie metadanych (query by example) lub na podstawie podobieństwa treści.
7. Wykonywane są reguły sprawdzające spełnienie założeń sprawdzających krzyżową zgodność dokumentów. Wynikiem ewaluacji jest wartość logiczna true/false. Wyliczenie tej wartości i umieszczenie w kontekście procesu umożliwia przekazanie procesu przez bramki decyzyjne i podjęcie odpowiednich działań w zależności od tego, czy dokument wejściowy jest prawidłowy czy nie. Wynikiem może być np. wprowadzenie prawidłowego dokumentu do systemu FK przez robota. W przypadku

wykrycia nieprawidłowego dokumentu może być wysyłane np. powiadomienie mailowe do użytkownika o zaistnieniu takiej sytuacji.

W końcowej fazie realizacji zadania przeprowadzono testy dotyczące weryfikacji krzyżowej dokumentów. Testy były przeprowadzone na próbce **200 par dokumentów Umowa i Faktura**. W pierwszym kroku przeprowadzona została ekstrakcja reguł z umów. Dla każdej umowy zostały zapisane w repozytorium wyekstrahowane reguły w formie interpretowanej przez walidator (format JSON). Następnie dla każdej faktury wyzwolono proces przetwarzania, w którym najpierw ekstrahowane są wartości atrybutów faktury. Następnie wyszukiwana jest powiązana z fakturą umowa i pobierane są związane z nią reguły. Następnie do ewaluatora przekazywane są odczytane wartości atrybutów i wyliczany jest wynik ewaluacji.

TP 86%

TN 2%

FP 4%

FN 8%

Do obliczenia statystyk przyjęto następujący podział:

- TP, True Positive: ewaluator zwraca prawdę i jest to wartość poprawna, dokumenty są zgodne.
- TN: True negative: ewaluator zwraca fałsz i jest to wartość poprawna, dokumenty nie są zgodne (najczęstszy powód to odchylenia wartości kwoty na fakturze i zbyt krótki termin płatności). Jest to najbardziej istotny biznesowo przypadek, który przekierowuje dokument na ścieżkę obsługi nieprawidłowych dokumentów.
- FP: False Positive: ewaluator zwraca prawdę ale dokumenty nie są zgodne. Takie przypadki zazwyczaj są związane z błędami na etapie samej ekstrakcji reguł i występują najczęściej w przypadku kiedy zestaw reguł nie jest kompletny (nie wyekstrahowano wszystkich reguł).
- FN: False Negative: ewaluator zwraca fałsz, ale dokumenty są zgodne. Jest to najczęściej spowodowane nieprawidłową wartością parametru reguły, np. odczytano inną wartość, niż jest faktycznie określona w dokumencie. Najczęściej jest to związane z błędami procesu OCR.

KAMIEŃ MIŁOWY:

Stworzono zestaw algorytmów umożliwiający automatyczną weryfikację krzyżową dokumentów co najmniej dla faktur i umów ze skutecznością 86% prawidłowo zweryfikowanych dokumentów na zbiorze testowym 200 par dokumentów.

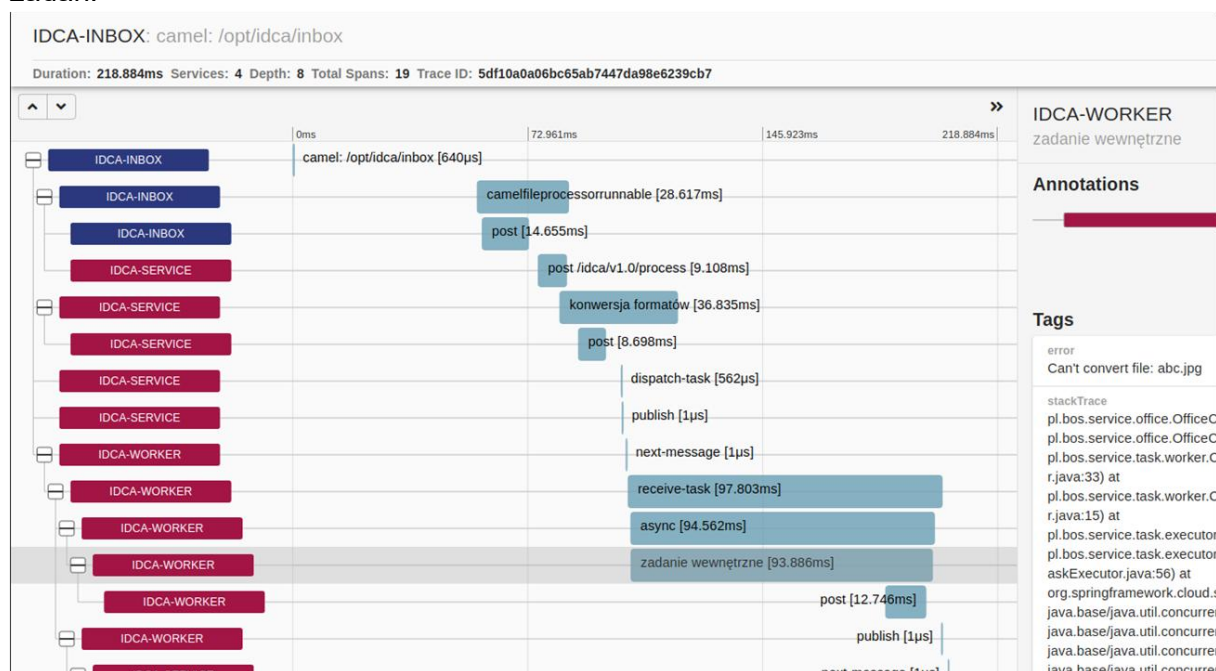
Działanie 3 (prace rozwojowe)

Zadanie nr 1. Opracowanie architektury wdrożeniowej opartej na mikro serwisach i konteneryzacji:

Podczas realizacji zadania stworzono środowisko docker do konteneryzacji serwisów oraz opracowana została architektura trawingu komunikatów, centralizacji logów i monitoringu serwisów. Monitoring serwisów został oparty na narzędziu Zipkin. Skalowanie warstwy przetwarzania dokumentów odbywa się przez zastosowanie niezależnych serwisów (workerów), które są połączone kolejką komunikatów (rabbitMQ).



Możliwe jest precyzyjne śledzenie wywołań między serwisami z pomiarem czasu wykonywania zadań.



Na poziomie wykonywania procesów biznesowych zaimplementowane zostało szczegółowe śledzenie wykonania poszczególnych zadań procesowych w dedykowanym narzędziu Tracer. Dzięki temu zapewniona jest pełna audytowalność procesów.

Tracer

Filtruj

Data od

Data do

Nazwa	Czas utworzenia	Czas procesu	Czas przetwarzania
Mail: Dokument F-K_k.malas@bosmanager.com_17.11.2020_13.05.16 z załącznikami	17.11.2020, 13.05	8,162 sekundy	429 sekundy

TraceId: 5fb3bc7ec384acf03ce67b6e9fbf28fa

Wejście: post /idca/v1.0/process — Mail: Dokument F-K_k.malas@bosmanager.com_17.11.2020_13.05.16 z załącznikami

Kroki procesu: 44

Nazwa	Czas utworzenia	Serwis	Czas procesu
Koniec procesu	13.05.27	idca-service	0 sekund
Mail: RE: Dokument F-K [automatyczna odpowiedź]	13.05.26	idca-worker	0 sekund
Autoresponder: Wiadomość sklasyfikowana jako korekta Faktury	13.05.26	idca-service	0,528 sekundy
Utworzono sprawę: Wiadomości	13.05.26	idca-worker	0 sekund
Utwórz Sprawę: Korekty Faktur -> Do weryfikacji	13.05.25	idca-service	1,24 sekundy
Utworzono: /opt/idca/WORKER/hyper/Faktury/2020/11/17/Korekt13.05.25 F-K_korekta FV.zip	13.05.25	idca-worker	0 sekund
Rodzaj faktury >> Korekta	13.05.25	idca-service	0 sekund
Utworzono zmienną: invoiceKind	13.05.25	idca-service	0 sekund
Utworzono zmienną: ocr	13.05.25	idca-service	0 sekund

Czas i status przetwarzania każdego dokumentu jest widoczny w tracerze.

Search

Name	Creation Time	Processing Time
FV-12489-12-2019.pdf	12/11/19, 4:09 PM	0.105 seconds
FV-12488-12-2019.pdf	12/11/19, 4:09 PM	1.1 seconds
20190329090708298-0.pdf	12/11/19, 4:04 PM	2 minutes, 0.747 seconds

Items per page: 10 1 - 3 of 3

Prawidłowo przetworzone dokumenty.

Search

2019.pdf

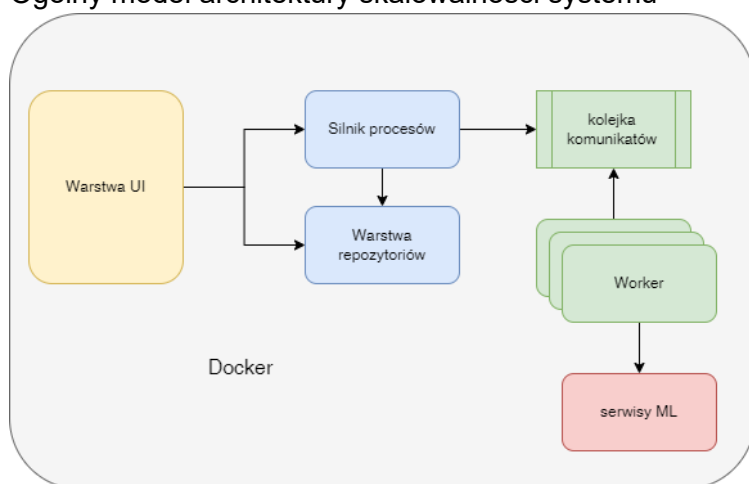
Name	Creation Time	Processing Time
FV-12489-12-2019.pdf	12/11/19, 4:09 PM	32.109 seconds
FV-12488-12-2019.pdf	12/11/19, 4:09 PM	23.901 seconds

Items per page: 10 1 - 2 of 2

Zadanie nr 2. Opracowanie formatów i metod wymiany danych między komponentami:

W ramach zadania zaimplementowano opracowane algorytmy w serwisach REST na platformie Spring. Potoki przetwarzania dokumentów i procesów biznesowych oparto na centralnym silniku procesów biznesowych, który przechowuje stan procesu dla każdego odrębnego dokumentu. Poszczególne zadania procesowe są delegowane do niezależnych serwisów (workerów) za pomocą komunikacji asynchronicznej opartej na wymianie komunikatów z pośrednictwem brokera komunikatów rabbitMQ.

Ogólny model architektury skalowalności systemu



Przykładowy zrzut komunikatu zawierającego dane dokumentu.

DEMO
<pre> { "width": 2480, "page": 1, "height": 3507 }, { "width": 2480, "page": 2, "height": 3507 }, { "width": 2480, "page": 3, "height": 3507 }, { "width": 2480, "page": 4, "height": 3507 }, { "width": 2480, "page": 5, "height": 3507 } }, "fullText": "Suma kontrolna: fd9f-2130-a4d7\n\nStrona: 1\nUMOWA O REALIZACJĘ ZADANIA PUBLICZNEGO* /\nKTÓREJ MOWA W ART. 16 UST.", "fields": [{ "score": 0.6, "bboxes": [{ "top": 825, "left": 1203, "bottom": 871, "page": 0, "right": 1467 }], "value": "23/0001421", "fieldId": "subject" }, { "score": 0.7, "bboxes": [{ "top": 1241, "left": 613, "bottom": 1283, "page": 0, "right": 896 }], "value": "31 maja 2023", "fieldId": "signedDate" }, { "score": 0.7, "bboxes": [{ "top": 2515, "left": 1085, "bottom": 2557, "page": 1, "right": 1289 }], "value": "55000,00", "fieldId": "agreementValue" }, { "score": 1, "bboxes": [{ "top": 3106, "left": 738, "bottom": 3148, "page": 5, "right": 1490 }], "value": "57 1020 4795 0000 9702 0277 8470", "fieldId": "Bank.account" }, { "score": 181.38364779874215, "bboxes": [{ "top": 2240, "left": 356, "bottom": 2283, "page": 0, "right": 634 }], "value": "33/BD0/2023", "fieldId": "Number" }] } </pre>

Zadanie nr 3. Opracowanie interfejsu użytkownika i implementacja prototypu:

W trakcie realizacji tego zadania wykonana została konsolidacja wytworzonych wcześniej komponentów systemu i wytworzony został interfejs użytkownika do elementów procesu wymagających interakcji z użytkownikiem.

Poniżej przedstawiono ekran służący do annotacji danych. Ekran ten na etapie przygotowywania danych treningowych był wykorzystywany do tworzenia opisów metadanych zgromadzonego korpusu dokumentów. Na etapie produkcyjnym może być wykorzystywany do weryfikacji danych i douczania systemu.

Umowa, przykład 1.

The screenshot shows a web application for creating contracts. On the left, there is a form with various input fields for contract details. On the right, there is a preview of the contract text, which includes the title, date, and parties involved.

Form Fields:

- Strona:** [input field]
- Umowa w przygotowaniu:** [checkbox]
- Kod kreacji:** [input field]
- Numer umowy:** 23/000-456
- Numer porządkowy (pusty):** [input field]
- Redakcja umowy:** [input field]
- Arka:** [input field]
- Osoby odpowiedzialne za realizację:** [input field]
- Kontaktem główny:** [input field]
- Pozostałe kontakty:** [input field]
- Data zawarcia umowy:** 01-06-2023
- Data rozpoczęcia umowy:** [input field]
- Data zakończenia obowiązywania:** [input field]
- Termin płatności:** [input field]
- Wartość netto:** 0.00
- Wartość brutto:** 30 000.00
- Dostępna wartość do wykorzystania podczas realizacji:** 30 000.00
- Oznaczenie wypowiadania:** [input field]
- Płatność ratowa:** [input field]
- Skrócony przedmiot umowy:** [input field]
- Przedmiot umowy:** [input field]

Contract Text Preview:

UMOWA O REALIZACJĘ ZADANIA PUBLICZNEGO
O KTOREJ MOWA W ART. 16 UST. 1 USTAWY Z DNIA 24 KWIEŹNIA
2003 R. O DZIAŁALNOŚCI POŻYTKU PUBLICZNEGO I O WOLONTARIACIE

BDO-4.524.256.2023.JG
CRU Nr 23/0001488

pod tytułem: KTO? NGO! Pozarządowy Szczecin | KONKURS 2023,
zawarta w dniu 01.06.2023 w Szczecinie,
między:
Gminą Miasto Szczecin, z siedzibą w Szczecinie 70-456, Plac Armii Krajowej 1, zwaną
dalej „Zleceniodawcą”, reprezentowaną przez: Daniela Wacinkiewicza – Zastępcę
Prezydenta Miasta Szczecin;
a
Fundacją Sektor 3, z siedzibą w 70-441 Szczecin, ul. Księcia Bogusława X 10/10 wpisaną
do Krajowego Rejestru Sądowego pod numerem 0000491200, zwaną dalej „Zleceniobiorcą”,
reprezentowaną przez:
1. Jakuba Szombkę – Dyrektora Fundacji,
2. Katarzynę Jurewic – Dyrektorkę Fundacji
zgodnie z wyciągiem z Krajowego Rejestru Sądowego, załączonym do niniejszej umowy,

Umowa, przykład 2.

The screenshot shows a web application for creating contracts. On the left, there is a form with various input fields for contract details. On the right, there is a preview of the contract text, which includes the title, date, and parties involved.

Form Fields:

- Strona:** [input field]
- Umowa w przygotowaniu:** [checkbox]
- Kod kreacji:** [input field]
- Numer umowy:** 23/000-456
- Numer porządkowy (pusty):** [input field]
- Redakcja umowy:** [input field]
- Arka:** [input field]
- Osoby odpowiedzialne za realizację:** [input field]
- Kontaktem główny:** [input field]
- Pozostałe kontakty:** [input field]
- Data zawarcia umowy:** 01-06-2023
- Data rozpoczęcia umowy:** [input field]
- Data zakończenia obowiązywania:** [input field]
- Termin płatności:** [input field]
- Wartość netto:** 0.00
- Wartość brutto:** 30 000.00
- Dostępna wartość do wykorzystania podczas realizacji:** 30 000.00
- Oznaczenie wypowiadania:** [input field]
- Płatność ratowa:** [input field]
- Skrócony przedmiot umowy:** [input field]
- Przedmiot umowy:** [input field]

Contract Text Preview:

z tytułu: „Zleceniobiorców: Agnieszka Wacinkiewicz, tel. adres poczty elektronicznej

§ 2
Sposób wykonania zadania publicznego

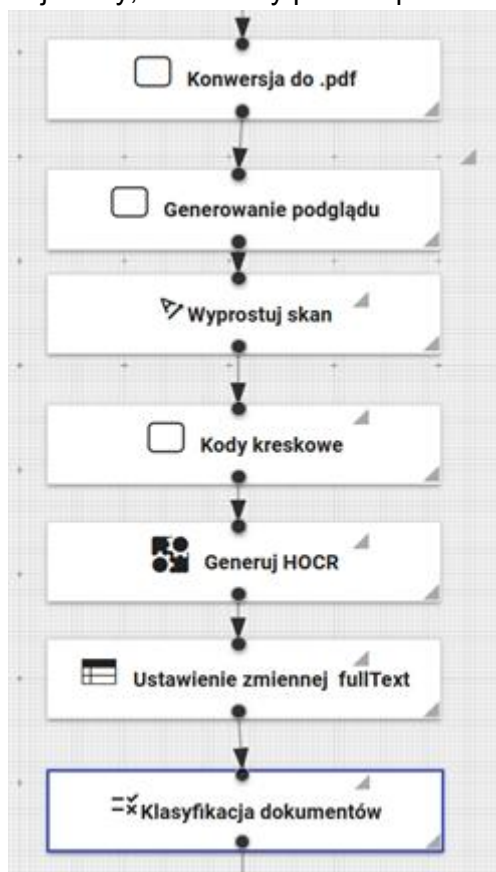
- Termin realizacji zadania publicznego ustala się:
od dnia 01.08.2023 r.
do dnia 31.12.2023 r.
- Termin poniesienia wydatków ustala się:
1) dla środków pochodzących z dotacji:
od dnia 01.08.2023 r.
do dnia 31.12.2023 r.;
2) dla innych środków finansowych:
od dnia 01.08.2023 r.
do dnia 31.12.2023 r.
- Zleceniobiorca zobowiązuje się wykonać zadanie publiczne zgodnie z ofertą, w terminie określonym w ust. 1.
- Zleceniobiorca zobowiązuje się do wykorzystania środków, o których mowa w § 3 ust. 1 zgodnie z celem, na jaki je uzyskał, i na warunkach określonych w niniejszej umowie. Dopuszcza się wydatkowanie uzyskanych przychodów, w tym także odsetek bankowych od środków przekazanych przez Zleceniodawcę, na realizację zadania publicznego wyłącznie na zasadach określonych w umowie. Niewykorzystane przychody Zleceniobiorca zwraca Zleceniodawcy na zasadach określonych w § 10.
- Wydatkowanie osiągniętych przychodów, w tym także odsetek bankowych od środków przekazanych przez Zleceniodawcę, z naruszeniem postanowień ust. 4 uznaje się za dotację pobraną w nadmiernej wysokości.

§ 3
Finansowanie zadania publicznego

- Zleceniodawca zobowiązuje się do przekazania na realizację zadania publicznego środków finansowych w wysokości 30 000,00 zł (słownie) trzydzieści tysięcy złotych 00/100, na rachunek bankowy Zleceniobiorcy:

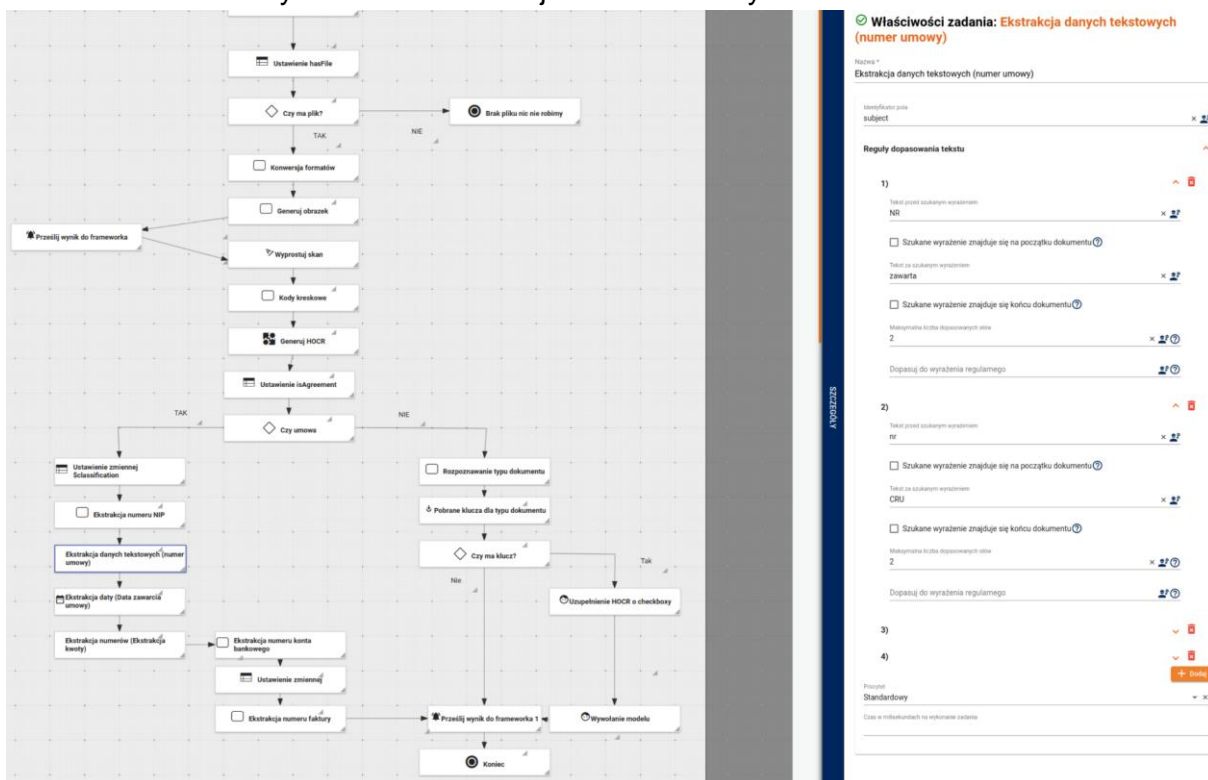
Jak było określone na etapie tworzenia koncepcji architektury całe rozwiązanie jest oparte na silniku procesów biznesowych, który służy do koordynacji zadań procesów przetwarzania dokumentów oraz powiązanych z nimi procesów biznesowych.

Wstępny etap przetwarzania jest podobny dla wszystkich rodzajów dokumentów. Jest to najniższy, techniczny poziom procesów.

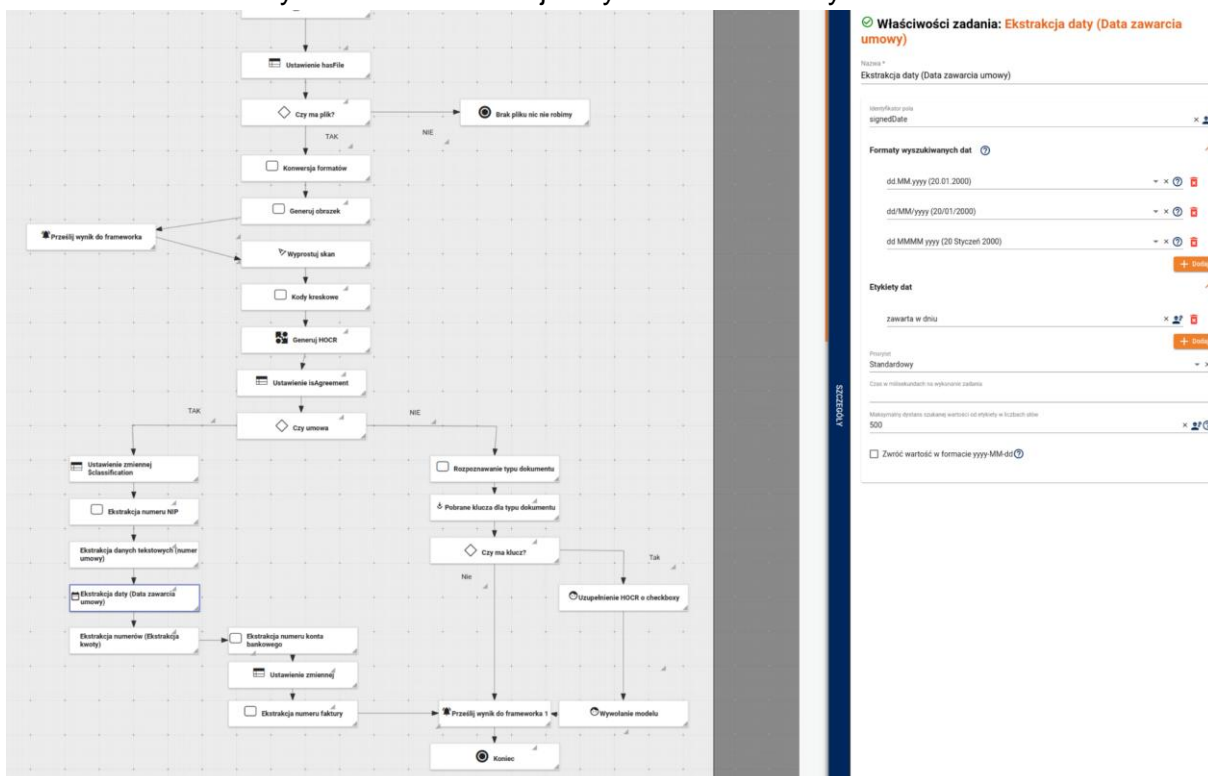


Konkretne rodzaje dokumentów są przetwarzane w dedykowanych procesach, które wykonują kolejne zadania wywołując wytrenowane wcześniej modele uczenia maszynowego, które są dostępne jako serwisy. Jest to pośredni poziom procesów odpowiedzialny za ekstrakcję i walidację danych.

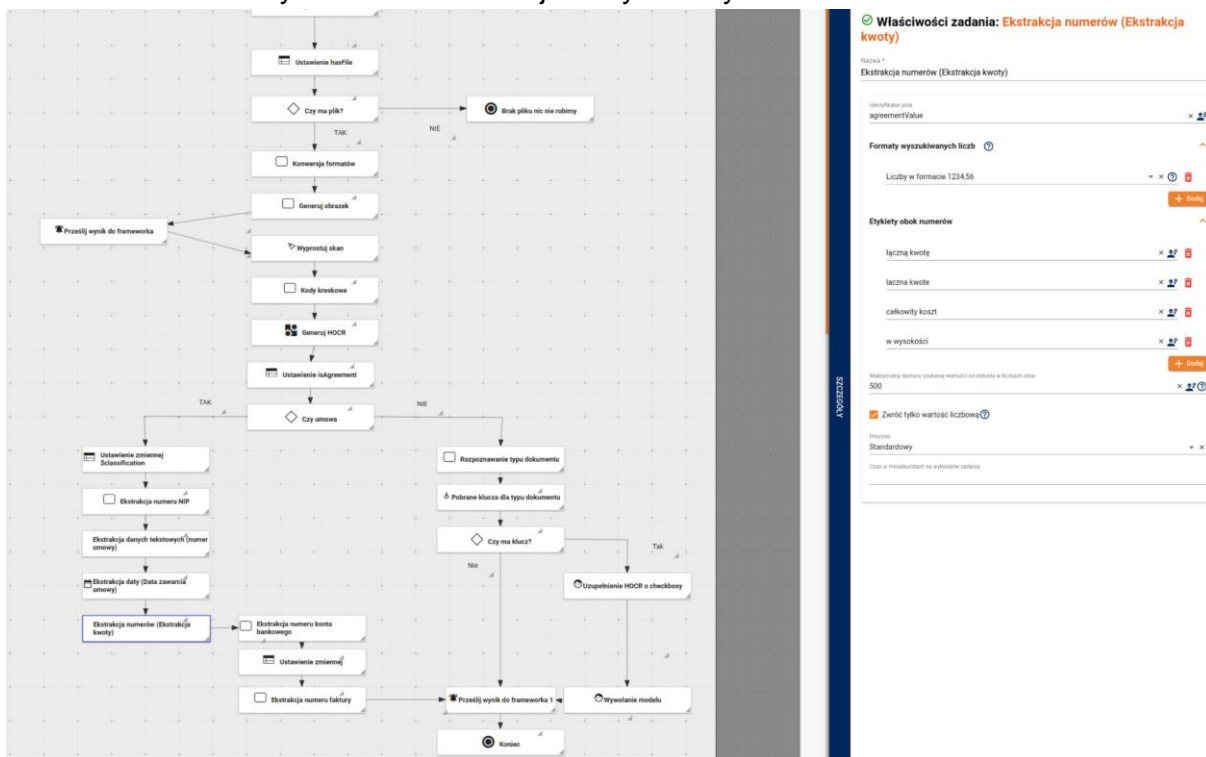
Przetwarzanie umowy - zadanie ekstrakcji numeru umowy



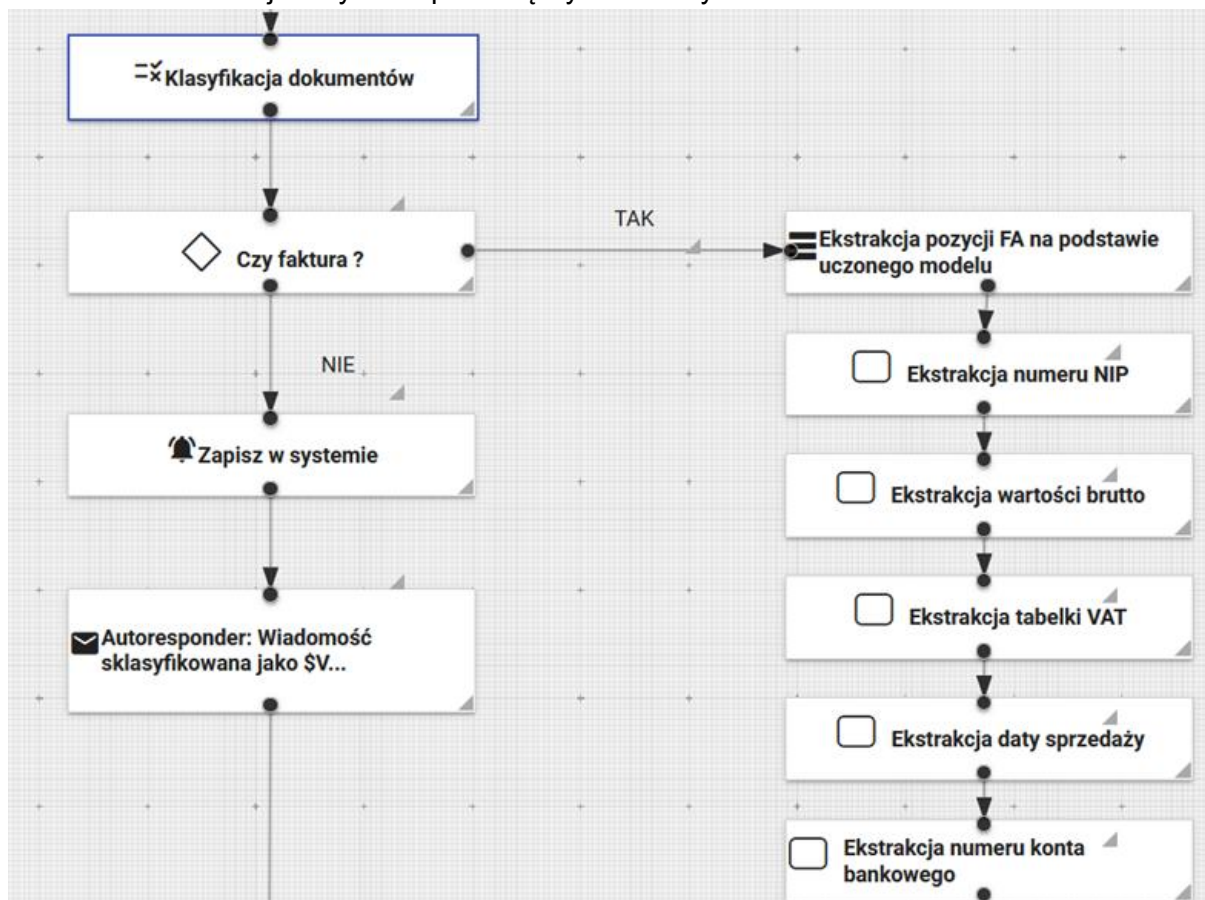
Przetwarzanie umowy - zadanie ekstrakcji daty zawarcia umowy



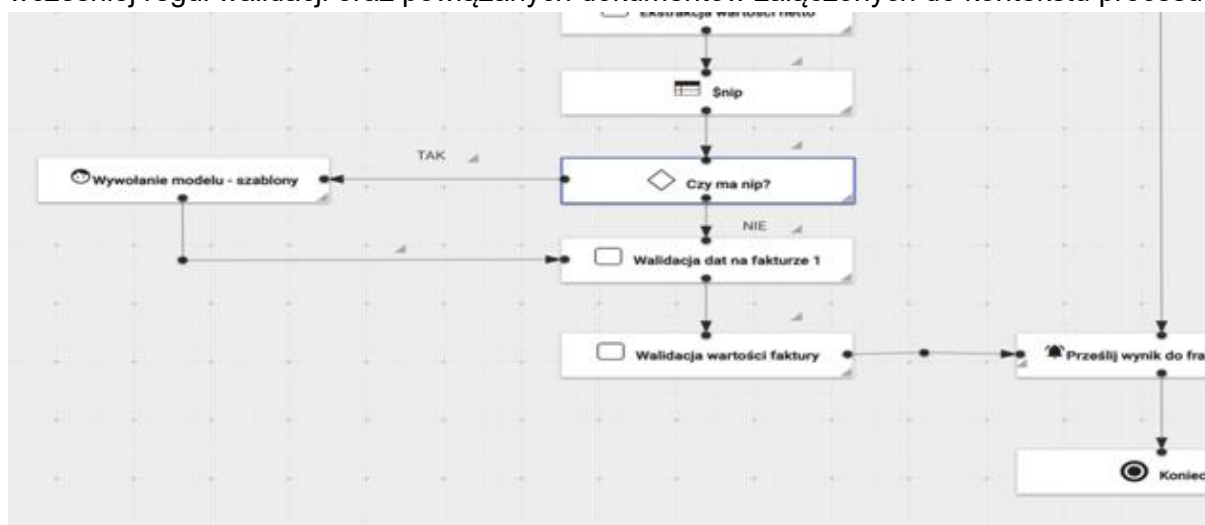
Przetwarzanie umowy - zadanie ekstrakcji kwoty umowy.



Dla dokumentów typu Faktura skonfigurowany jest dedykowany proces, który rozpoczyna się również od ekstrakcji danych za pomocą wytrenowanych modeli.

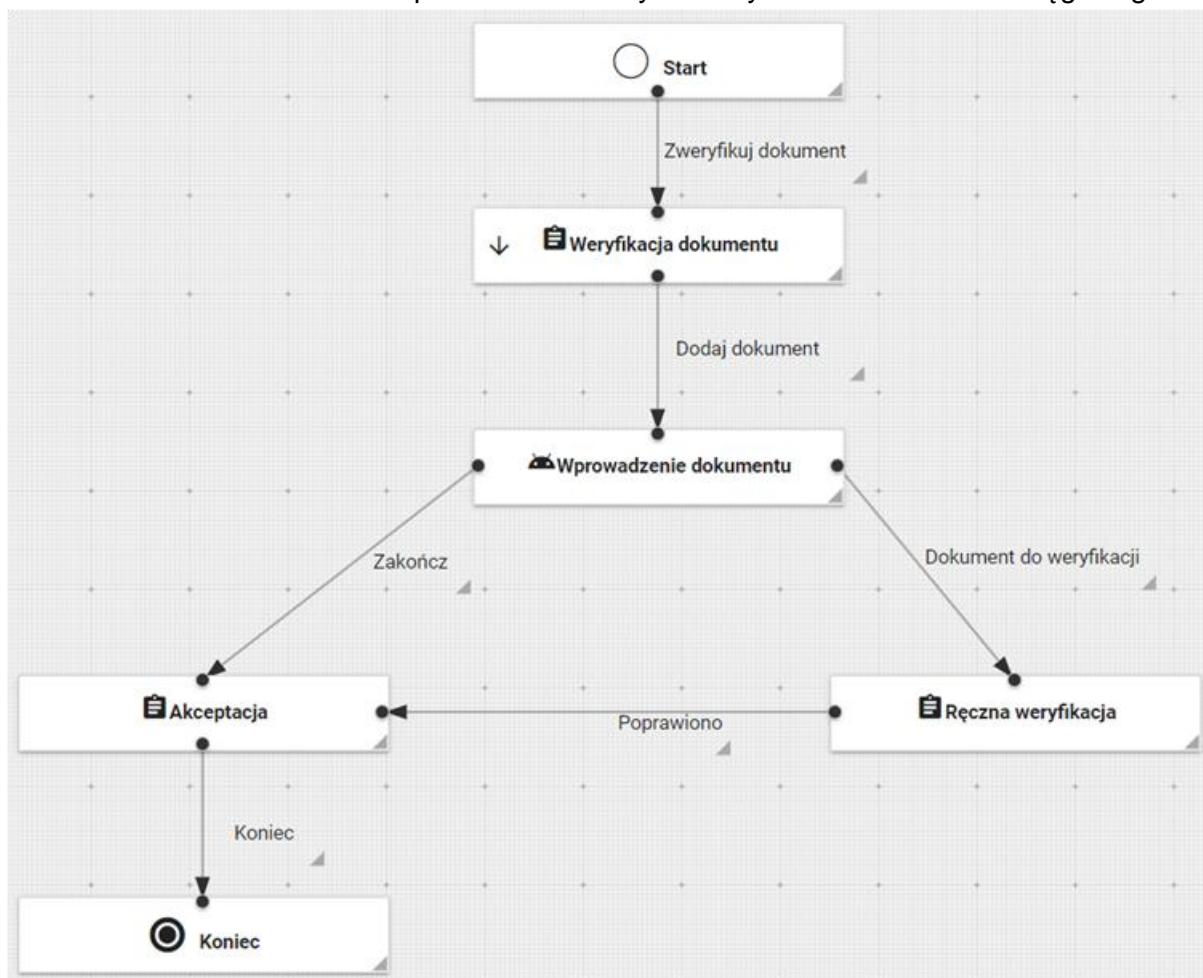


Następnie proces przechodzi do walidacji wartości atrybutów na podstawie utworzonych wcześniej reguł walidacji oraz powiązanych dokumentów załączonych do kontekstu procesu.

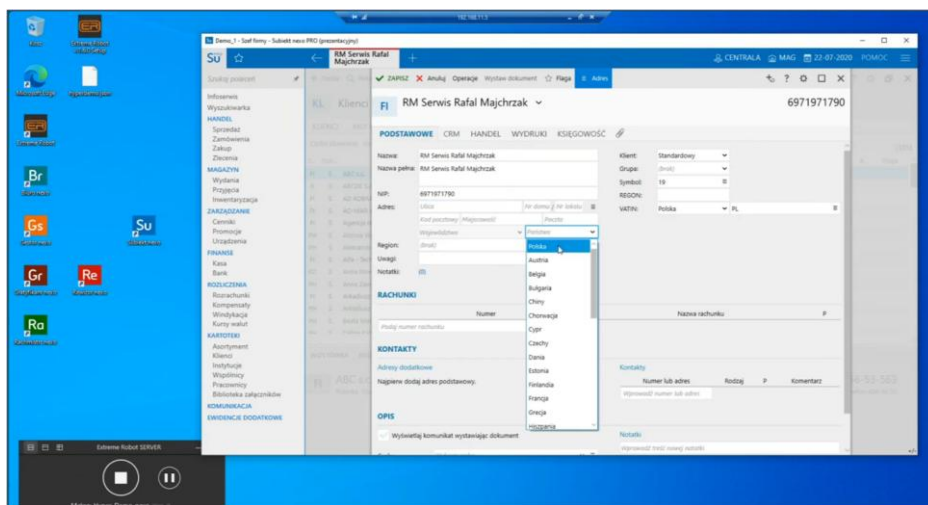


Na najwyższym poziomie wykonywany jest proces biznesowy, który obejmuje całość operacji:

- przetworzenie dokumentu wejściowego
- ekstrakcję danych
- walidację danych
- wykonanie działań na poprawnie zweryfikowanych dokumentach np. poprzez uruchomienie robota i wprowadzenie danych do systemu finansowo - księgowego.



Robot automatycznie wprowadza fakturę do systemu FK.



W wyniku przeprowadzonych prac projektowych powstał prototyp systemu zdolny do automatycznej walidacji strumienia nowych dokumentów z założoną wydajnością, co było przedmiotem kamienia milowego.

W raporcie przedstawiono próbkę wyników czasu przetwarzania dokumentów w systemie z punktu widzenia narzędzia monitorującego przebieg procesów:

Search
2019.pdf

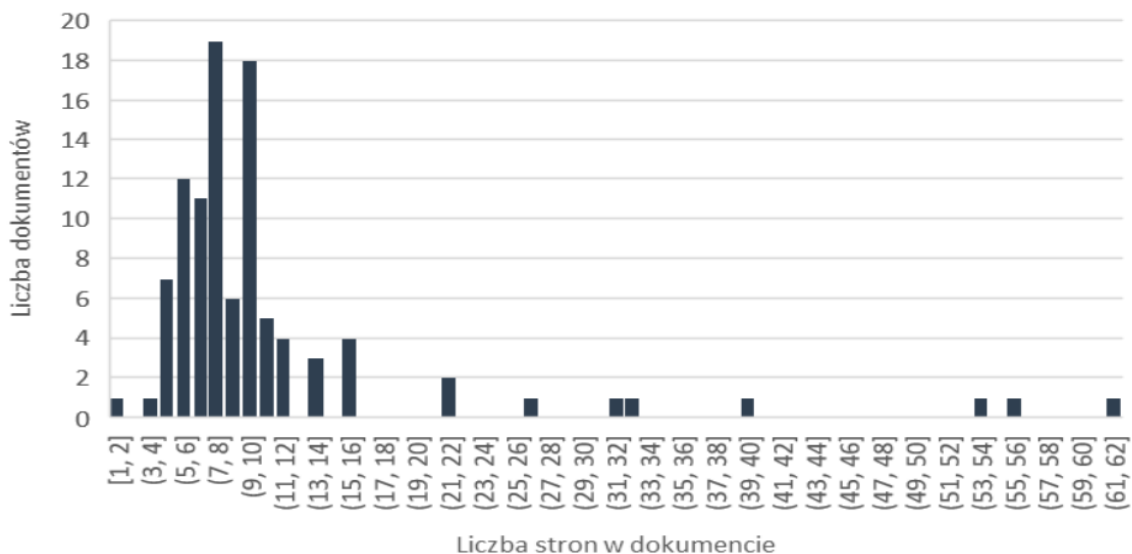
Name	Creation Time ↓	Processing Time
FV-12489-12-2019.pdf	12/11/19, 4:09 PM	32.109 seconds
FV-12488-12-2019.pdf	12/11/19, 4:09 PM	23.901 seconds

Items per page: 10 1 - 2 of 2 |< < > >|

Ogólna statystyka pozyskana z zebranych danych dla próbki 50 dokumentów wynosi:

- Średni czas przetwarzania: 28,5 sekundy
- Odchylenie standardowe: 12,3 sekundy

Duże odchylenie standardowe jest związane z bardzo dużą rozbieżnością ilości stron w dokumentach (od 1 do 62 stron). Jednak rozkład liczby stron pokazuje, że nie ma sensu optymalizować rozwiązania pod kątem rzadko występujących dokumentów o liczbie stron większej niż 10.



KAMIEŃ MIŁOWY:

Potwierdzono zatem osiągnięcie kamienia milowego dla Zadania 3 - Opracowanie prototypu systemu krzyżowej walidacji dokumentów:

Uzyskano wdrażalny prototyp systemu zdolny do automatycznej walidacji strumienia nowych dokumentów w czasie zbliżonym do rzeczywistego z opóźnieniem nie większym niż 28,5s/dokument.