Streszczenie

Rekomendacje artykułów opisujących produkty w serwisach e-commerce

Tematyka niniejszej pracy skupia się wokół zagadnień określania podobieństwa semantycznego pomiędzy dokumentami tekstowymi i rekomendacji dokumentów podobnych. Szczegółowy problem pochodzi z internetowego serwisu aukcyjnego *Allegro*, który posiada dział artykułów opisujących produkty dostępne na platformie. W dziale tym funkcjonuje system rekomendacji podobnych artykułów tekstowych w oparciu o ich treść. Celem pracy jest zbadanie możliwości usprawnienia działania istniejącego systemu rekomendacji wykorzystując metody semantycznej analizy tekstu.

W niniejszej pracy adaptuję dostępne metody określania podobieństwa pomiędzy dokumentami tekstowymi do powyższego problemu, wprowadzam miary umożliwiające ocenę działania tych metod oraz dokonuję analizy możliwości ich wykorzystania w rzeczywistym systemie.
Słowa kluczowe: rekomendacje, przetwarzanie języka naturalnego, osadzanie słów, semantyka, allegro

Abstract

Content-based recommendations in e-commerce services

The subject of this paper focuses on issues of determining the semantic similarity between text documents and the recommendation of similar documents. A detailed problem comes from the on-line auction site *Allegro*, which has a section of articles describing the products available on the platform. This section offers a recommendation system for similar textual articles based on their content. The aim of this paper is to investigate the possibility of improving the existing recommendation system using semantic text analysis methods.

In this paper, I adapt the state-of-the-art methods for determining the similarity between text documents to the above problem, I introduce measures to evaluate the performance of these methods and analyze the possibilities of using them in the real system.

Keywords: recommendations, natural language processing, word embedding, semantics, allegro

Łukasz Dragan
Nr albumu 254179

Warszawa, dnia .....

Oświadczenie

Oświadczam, że pracę magisterską pod tytułem „Rekomendacje artykułów opisujących produkty w serwisach e-commerce", której promotorem jest dr inż. Anna Wróblewska wykonałem samodzielnie, co poświadczam własnoręcznym podpisem.

.....

Łukasz Dragan

Spis treści

Wstęp

Systemy rekomendacji są powszechnym elementem wielu serwisów internetowych. Sprawdzają się na takich polach, jak polecanie produktów w sklepie czy rekomendacje ofert pracy. Dają użytkownikowi poczucie indywidualnego traktowania przez serwis internetowy dopasowujący niejako zawartość swoich stron do konkretnego użytkownika. Pozwala to użytkownikowi na bardziej efektywne korzystanie z serwisu oraz może prowadzić do większego zaangażowania ze strony użytkownika i przywiązania do serwisu. Systemy rekomendacji dają korzyść zarówno użytkownikowi jak i właścicielowi serwisu internetowego.

Celem niniejszej pracy magisterskiej jest analiza możliwości usprawnienia istniejącego systemu rekomendacji w oparciu o adaptację istniejących metod wyszukiwania semantycznego podobieństwa pomiędzy dokumentami tekstowymi. Rzeczony system rekomendacji istnieje w internetowym serwisie e-commerce *Allegro* w dziale artykułów tekstowych o tematyce związanej z produktami dostępnymi za pośrednictwem serwisu. System ma na celu zarekomendowanie użytkownikowi artykułów o tematyce podobnej do tego, który znajduje się na stronie aktualnie odwiedzanej przez użytkownika.

W swojej pracy badam możliwość użycia istniejących metod semantycznej analizy tekstu w odniesieniu do opisanego problemu. Badane metody to: *Latent Semantic Analysis*, *Latent Dirichlet Allocation*, *Word2vec*, *GloVe* oraz *FastText*. Jakość działania tych metod porównuję poprzez samodzielnie opracowane metody ewaluacji.

Podczas prowadzenia badań stworzyłem szereg skryptów przetwarzających dane i wykorzystujących implementacje opisywanych w tej pracy metod. Opis użytych narzędzi programistycznych i bibliotek zawarłem w załączniku 1 do niniejszej pracy. Stworzone skrypty umieszczam na płycie CD dołączonej do pracy. Opis zawartości płyty znajduje się w załączniku 2.

System rekomendacji artykułów tekstowych w *Allegro*

Allegro jest największą [25] działającą na rynku polskim platformą aukcyjną on-line. Posiada ponad 20 milionów zarejestrowanych klientów. Każdego dnia za pośrednictwem *Allegro*

sprzedaje się ponad 870 tysięcy przedmiotów. Firma zatrudnia 1300 pracowników [1]. Serwis umożliwia użytkownikom wystawianie na sprzedaż oraz kupno przedmiotów poprzez mechanizm licytacji lub natychmiastowego zakupu.

Oprócz głównej części serwisu odpowiedzialnej za transakcje, *Allegro* posiada dział zajmujący się publikacją artykułów opisujących produkty wystawiane za pośrednictwem serwisu. Ma to na celu pomoc użytkownikom przy wyborze interesującego ich produktu.

Po to, aby zachęcić użytkowników do zapoznania się z treścią kolejnych artykułów, zastosowany został tu system rekomendacji przyporządkowujący danemu artykułowi listę powiązanych artykułów. Kryterium mówiącym, czy artykuły są powiązane jest tutaj jedynie treść samych artykułów, a nie wcześniejsze zachowanie użytkownika.

Rysunek 0.1: Widok strony internetowej zawierającej jeden z artykułów serwisu *Allegro* [33].

Od serwisu *Allegro* otrzymałem kopię ok. 20000 zserializowanych artykułów dostępnych na stronach serwisu. Pojedynczy artykuł składa się z głównej zawartości tekstowej oraz metadanych. W celu otrzymania wszelkich danych od firmy *Allegro* wymagane było, abym podpisał umowę, w której zobowiązuję się do nieujawniania żadnych danych, które otrzymałem. Stąd opisy danych, na których pracuję, zawarte w tej pracy nie wnikają w ich szczegóły i nie odbiegają od informacji publicznie dostępnych za pośrednictwem strony pod adresem `https://allegro.pl/artykuly`.

Aktualnie w rzeczonym dziale serwisu Allegro istnieje system rekomendacyjny, który opiera się o wyszukiwanie podobnych artykułów tekstowych za pomocą silnika *Elasticsearch* [8]. Metoda ta wykorzystuje słowa kluczowe przypisane do każdego artykułu przez autora. W swojej pracy staram się porównać wyniki działania dotychczasowej metody z metodami semantycznej analizy tekstu, które potrafią wykryć podobieństwo pomiędzy artykułami bazując jedynie na ich treści, bez potrzeby dołączania żadnych metadanych. Pomyślna próba zastosowania metod semantycznych pozwoliłaby na dokładniejsze dopasowanie podobnych artykułów w oparciu być może o pewne ukryte cechy semantyczne nieosiągalne dla silnika wyszukiwania tekstowego, jakim jest *Elasticsearch*. Bardziej szczegółowego opisu silnika *Elasticsearch* dokonuję w kolejnym rozdziale.

Struktura pracy

W rozdziale 1 wprowadzam do zagadnienia systemów rekomendacji oraz dokonuję przeglądu metod semantycznej analizy tekstu, które mogą zostać zastosowane w celu określenia podobieństwa pomiędzy dokumentami tekstowymi.

Następnie w rozdziale 2 dokonuję opisu konkretnego problemu, jakim jest generacja rekomendacji artykułów tekstowych w serwisie *Allegro*. Opisuję dane otrzymane z serwisu oraz kolejne etapy ich wstępnego przetwarzania, aby nadawały się do zaaplikowania do nich wybranych metod.

Dalej, w rozdziale 3 opisuję stworzone i zastosowane później metody ewaluacji wyników.

Następnie, w rozdziale 4, dokonuję opisu testów: jakie metody i w jaki sposób testuję.

W rozdziale 5 opisuję wyniki przeprowadzonych eksperymentów.

Ostatecznie w rozdziale 6 dokonuję podsumowania przeprowadzonych badań i rozważam kierunki dalszych prac w tej dziedzinie.

Uwagi

W celu uniknięcia nieporozumień należy podkreślić różnicę pomiędzy znaczeniami słowa „artykuł", które może oznaczać zarówno tekst publicystyczny, literacki lub naukowy jak i rzecz, która jest przedmiotem handlu [32]. W niniejszej pracy skupiam się na rekomendacjach artykułów tekstowych, stąd używam pierwszego znaczenia (chyba, że inne znaczenie jest wyraźnie zaznaczone).

Z racji szybkiego rozwoju szeroko pojętych metod sztucznej inteligencji, w tym metod przetwarzania języka naturalnego, wiele z pojęć opracowanych głównie w firmach, uczelniach i instytutach anglosaskich nie ma jeszcze swoich polskich odpowiedników. Często również mimo istnienia polskiego odpowiednika pojęcie w języku angielskim jest tak popularne, że używanie polskiego odpowiednika powodowałoby niepotrzebny zamęt. Stąd w wielu przypadkach stosuję angielskie pojęcia, dołączając do nich w nawiasach, przy ich wprowadzaniu, ich polskie odpowiedniki. Dalej na ogół posługuję się skrótami nazw angielskich — forma ta jest najszerzej przyjęta.

Przegląd wiedzy z zakresu tematyki pracy

W swojej pracy dokonuję adaptacji metod przetwarzania języka naturalnego na potrzeby generowania rekomendacji artykułów tekstowych w oparciu o ich treść. W niniejszym rozdziale dokonuję przeglądu znanych metod z obszaru tematyki pracy dyplomowej, skupiając się szczególnie na nowo powstałych metodach wektorowej reprezentacji słów, które cieszą się obecnie dużym zainteresowaniem środowisk naukowych oraz firm komercyjnych.

Dokonuję krótkiego wprowadzenia do zagadnienia systemów rekomendacji oraz systemów wyszukiwania. Następnie wykonuję chronologiczny przegląd metod liczbowej, ciągłej reprezentacji słów zaczynając od trywialnych metod zliczania słów (*bag-of-words*, *TF-IDF*), przechodząc przez metody wykorzystujące koncepcję tematów (*latent semantic analysis*, *latent Dirichlet allocation*) i kończąc na głośnych ostatnio metodach osadzania słów w przestrzeni wektorowej (*Word2vec*, *GloVe*, *FastText*). Przy zarysie historycznym korzystam m.in. z artykułu [4].

Systemy rekomendacji

Systemy rekomendacji to narzędzia i techniki mające na celu zasugerowanie użytkownikowi przedmiotów. Sugestie te odnoszą się do różnych procesów podejmowania decyzji takich jak np.: które artykuły kupić, jakiej muzyki słuchać, czy też które wiadomości czytać. „Przedmiot" jest tutaj ogólnym pojęciem oznaczającym coś, co system poleca użytkownikowi [26].

Przy wciąż wzrastającej ilości danych użytkownicy serwisów internetowych często nie są w stanie dotrzeć do informacji, która ich interesuje. Jest to pole do rozwoju zautomatyzowanych systemów rekomendacyjnych polecających użytkownikom treści, które mogą ich zainteresować. Działanie takiego systemu daje zysk zarówno użytkownikowi, pozwalając mu dotrzeć do informacji, której mógłby samodzielnie nie odszukać, albo wręcz nie wiedzieć, iż taka informacja istnieje, jak i właścicielowi serwisu internetowego, któremu zależy, by przyciągnąć do siebie użytkowników, aby ci w jak największym stopniu korzystali z jego usług.

Sposoby działania systemów rekomendacji można podzielić na różne warianty, spośród których wyodrębnić można dwa najszerzej używane. Są to: filtrowanie kolaboratywne i filtrowanie oparte na treści.

Filtrowanie kolaboratywne

Filtrowanie kolaboratywne (*collaborative filtering*) opiera się na spostrzeżeniu, iż użytkownicy o podobnych preferencjach zachowują się podobnie. Stąd, jeżeli użytkownik zachowuje się podobnie do zaobserwowanej wcześniej grupy użytkowników, można przewidzieć jego preferencje na podstawie zachowań owej grupy. Istotną zaletą tej metody jest fakt, iż nie zależy ona od dziedziny, w której ulokowany jest system rekomendacji (w przeciwieństwie do rekomendacji opartych na treści), a jedynie od zachowań użytkowników [27].

Filtrowanie oparte na treści

W filtrowaniu opartym na treści (*content-based filtering*) przedmioty polecane użytkownikowi zależą od innych przedmiotów, na temat których stwierdzono, że użytkownik się nimi interesuje. Mogą się one opierać np. na podobieństwie przedmiotów: jeżeli użytkownik „lubi" przedmiot A, który jest podobny do przedmiotu B, to można spodziewać się, że również przedmiot B zainteresuje użytkownika. Technika ta jest mocno zależna od dziedziny rekomendowanych przedmiotów, gdyż wymaga wprowadzenia pewnej miary podobieństwa między nimi. Stąd jest trudniejsza do zastosowania, ale daje też możliwości nieosiągalne dla filtrowania kolaboratywnego [28].

Celem niniejszej pracy jest zbadanie metod sugerujących użytkownikowi artykuły podobne do aktualnie odwiedzanego, co wprost wiąże się z metodami używanymi w technice filtrowania opartego na treści.

Systemy wyszukiwania

Systemy wyszukiwania projektuje się w celu znajdywania informacji przechowywanych w systemie komputerowym. Wyniki wyszukiwania są zazwyczaj prezentowane w postaci listy, której elementy uszeregowane są od najbardziej związanych z szukaną frazą: słowami kluczowymi lub wyrażeniem w języku naturalnym. Celem systemu jest ograniczenie czasu poświęcanego przez użytkownika na samodzielnym przeszukiwaniu systemu [6].

W ujęciu ogólnym systemy wyszukiwania mają na celu sugerowanie tego, co użytkownik chciałby otrzymać. Natomiast systemy rekomendacji mają sugerować przedmioty potrzebne użytkownikowi nawet, jeżeli potrzeby te nie zostały bezpośrednio wyrażone.

Przykładem systemu wyszukiwania jest silnik *Elasticsearch* używany m.in. w *Allegro*. Metoda generowania rekomendacji obecnie wykorzystywana w serwisie opiera się właśnie o zapytanie do usługi *Elasticsearch* [8] wykorzystujące słowa kluczowe manualnie dołączone do artykułów. *Elasticsearch* jest popularnym silnikiem wyszukiwania tekstu opartym o indeks *Lucene* [18]. Działa w architekturze rozproszonej a komunikacja z nim następuje poprzez protokół *HTTP* i format *JSON*. Umożliwia on efektywne przechowywanie dokumentów tekstowych oraz efektywne ich wyszukiwanie.

Apache Lucene jest biblioteką napisaną w języku *Java* służącą do wyszukiwania tekstu, która w tym celu wykorzystuje mechanizm odwróconego indeksu. Zasada działania biblioteki polega na stworzeniu słownika ze wszystkich (odpowiednio wstępnie przetworzonych) słów dokumentów przeznaczonych do wyszukiwania. Następnie na bazie owego słownika tworzony jest odwrócony indeks. Każdemu ze słów przypisywana jest lista dokumentów, które zawierają to słowo. Pozwala to przyspieszyć proces wyszukiwania, gdyż w poszukiwaniu pojedynczego słowa biblioteka nie przeszukuje całego zbioru dokumentów, a jedynie słownik, który na ogół jest wielokrotnie krótszy.

Zaletą silnika *Elasticsearch* są jego wydajność, skalowalność, niezawodność i prostota użytkowania, co przekłada się na jego dużą popularność wśród np. serwisów internetowych [9].

Wadą metody jest to, że ogranicza się ona do wyszukiwania tekstowego pomijając aspekt semantyczny. Stwarza to trudności np. przy wyszukiwaniu synonimów lub homonimów.

Techniki przetwarzania języka naturalnego

Oparcie rekomendacji jedynie na treści artykułu wymaga zagłębienia się w tematykę analizy i przetwarzania języka naturalnego, wszak właśnie w języku naturalnym, zrozumiałym dla człowieka (polskim) pisane są owe artykuły. Język naturalny z powodu swojego niskiego stopnia sformalizowania nie jest niestety wprost zrozumiały dla maszyn. W związku z tym konieczne staje się tu użycie technik przetwarzania języka naturalnego (*natural language processing, NLP*), które to pozwalają wyodrębnić z tekstu pewne cechy, na bazie których maszyna obliczeniowa przy pomocy odpowiednich algorytmów jest w stanie określić podobieństwo pomiędzy dokumentami. W poniższych paragrafach dokonuję przeglądu technik matematycznej reprezentacji dokumentów pisanych w języku naturalnym. Warto wspomnieć, iż dziedzina ta bardzo dynamicznie się rozwija, a część z opisywanych metod zostało stworzonych na przestrzeni ostatnich kilku lat, czy wręcz miesięcy.

W celu formalizacji, w dalszych opisach stosowanych metod stosuję następujące oznaczenia:

- korpus C: zbiór dokumentów d_i,
- dokument d: skończony ciąg słów w_i,
- słowo w: skończony ciąg znaków c_i,
- słownik V zbudowany na korpusie C: zbiór wszystkich unikalnych słów korpusu C, $V = \{w \mid \exists_{d \in C}\ w \in d\}$.

Techniki podstawowe

Bag-of-words

Bag-of-words (worek słów; w dalszej części pracy używam skrótu *BOW*) [12] jest jedną z pierwszych koncepcji reprezentacji tekstu jako zbioru zawartych w nim słów w postaci wektorów. Metoda nie bierze pod uwagę kolejności słów w tekście, lecz liczbę ich wystąpień. Technika ta przypisuje każdemu słowu v_i ze słownika V zbudowanego na korpusie C wektor długości $|V|$ o wartości 1 na i-tym miejscu i wartościach 0 na pozostałych miejscach. Wybrany fragment tekstu należącego do korpusu można wtedy traktować jako sumę wektorowych reprezentacji słów należących do niego.

Istotną zaletą reprezentacji wektorowej dokumentów jest możliwość zdefiniowania miary odległości pomiędzy dokumentami (np. miara kosinusowa opisana później) odzwierciedlającej ich podobieństwo.

Technika *BOW* jest stosunkowo prosta, lecz jej wadą jest traktowanie każdego słowa z jednakową wagą. Pewne słowa (np. „i", „lub", „o") występują bardzo często, lecz ich wkład w znaczenie całego dokumentu jest marginalny. Kolejną wadą jest wysoka wymiarowość otrzymywanych wektorów równa wielkości słownika.

Stąd powstały bardziej zaawansowane techniki uwzględniające istotność słów dla znaczenia całego dokumentu. Mimo to metoda *BOW* jest często wykorzystywana w bardziej zaawansowanych technikach *NLP*, a także przy przetwarzaniu wstępnym tekstu przy okazji zabiegów usunięcia słów stopu oraz lematyzacji stosowanych w celu zmniejszenia wymiarowości przestrzeni wektorowej słów.

TF-IDF

Term frequency — inverse document frequency (ważenie częstością termów — odwrotna częstość w dokumentach; w dalszej części pracy posługuję się skrótem *TF-IDF*) [30] jest metodą reprezentacji tekstu jako zbioru słów przy jednoczesnym uwzględnieniu wagi słów, która zależy od częstości występowania słowa w korpusie.

Wartość *TF-IDF* słowa w_i w dokumencie d_j wyraża się wzorem 1.1:

$$tfidf_{ij} = tf_{ij} * idf_i,\ tf_{ij} = \frac{n_{ij}}{\sum_k n_{kj}},\ idf_i = log\frac{|D|}{|d : w_i \in d|}, \tag{1.1}$$

gdzie tf_{ij} — *term frequency* — to liczba wystąpień słowa w_i w dokumencie d_j podzielona przez liczbę słów dokumentu d_j. Natomiast idf_i — *inverse document frequency* — to liczba dokumentów w korpusie podzielona przez liczbę dokumentów zawierających przynajmniej jedno wystąpienie słowa w_i.

Dokumenty reprezentowane są tu jako wektory, składające się z wag słów występujących w każdym z nich. *TF-IDF* przechowuje informację o częstotliwości występowania słów biorąc przy tym pod uwagę istotność znaczenia słowa lokalnego w stosunku do jego znaczenia w kontekście całego zbioru dokumentów. W tej technice słowa występujące rzadko są premiowane względem słów pospolitych. Wadą metod tej i poprzedniej jest postać wektorów reprezentujących słowa: są to na ogół rzadkie wektory dużej wymiarowości.

Semantyka dystrybucyjna

Kolejne, bardziej zaawansowane, omawiane tu metody opierają się na tzw. *distributional hypothesis* — hipotezie zakładającej, że słowa występujące w tym samym kontekście niosą ze sobą podobne znaczenie [12][10]. Sprzyja to zastosowaniu metod algebry liniowej jako narzędzia obliczeniowego oraz sposobu reprezentacji tekstu. Podstawowe podejście polega na zgromadzeniu informacji o rozkładzie słów w dokumentach w postaci wielowymiarowych wektorów, a następnie wyodrębnieniu podobieństw pomiędzy tymi wektorami, które świadczyłyby o pewnych powiązaniach między reprezentowanymi słowami.

Modelowanie tematów

Analiza rozkładu słów w dokumentach tekstowych pozwala na wyodrębnienie podobieństw między słowami pod kątem: ich znaczenia (podobieństwo tematu słowa), ich osadzenia w stosunku do innych typów słów, czy też ich struktury wewnętrznej. Dwie istotne metody, które opisuję w niniejszej pracy: *latent semantic analysis* [7] oraz *latent Dirichlet allocation* [3] zakładają istnienie abstrakcyjnych niejawnych (*latent*) tematów, do których można przydzielić słowa wchodzące w skład korpusu.

Latent semantic analysis

Latent semantic analysis (ukryta analiza semantyczna, *LSA*) [7], znane również jako *latent semantic indexing* (ukryte indeksowanie semantyczne; w dalszej części pracy używam skrótu *LSI*)

dokonuje transformacji każdego dokumentu w wektor dł. $|V|$ posiadający na i-tym miejscu wagę *TF-IDF* i-tego słowa ze słownika. W ten sposób tworzona jest rzadka macierz: kolumny reprezentują dokumenty, a wiersze reprezentują unikalne słowa. W celu identyfikacji istotnych cech tej macierzy dokonuje się rozkładu według wartości osobliwych (*singular value decomposition* [11], *SVD*), który jest techniką redukcji wymiarowości. Celem użycia *SVD* jest redukcja liczby wierszy macierzy dla wydajniejszych dalszych obliczeń numerycznych oraz pozbycie się szumów, utrzymując jednocześnie podobieństwa pomiędzy kolumnami. Ostatecznie uzyskuje się macierz przynależności tematów do dokumentów, gdzie wiersze odpowiadające tematom można interpretować jako kombinacje pierwotnych wierszy-słów o podobnym znaczeniu. Np. *{(„samochód"), („ciągnik"), („jezdnia")} → {(1.3452 * „samochód" + 0.2828 * „ciągnik" + 0.3 * „jezdnia")}*. Wymiar uzyskiwanej macierzy jest ustalany za pomocą hiperparametru, który oznacza liczbę tematów. Używając uzyskanej macierzy, podobieństwo pomiędzy kolumnami-dokumentami obliczane jest wykorzystując odległość kosinusową. Metoda *LSA* łagodzi problem synonimów poprzez scalanie podobnych słów w jeden temat. Niweluje również problem homonimów, włączając je częściowo w skład różnych tematów. Niemniej jednak poprzez arbitralne ustalanie hiperparametru odpowiedzialnego za liczbę tematów część semantycznie odrębnych tematów może zostać wchłonięta przez inne lub też rozbicie na tematy może być zbyt „drobne" nie wykorzystując w pełni semantycznych powiązań.

Latent Dirichlet allocation

Latent Dirichlet allocation(ukryta alokacja Dirichleta; dalej używam skrótu *LDA*) [3] jest techniką automatycznego wykrywania niejawnych (*latent*) tematów zawartych w dokumentach przy użyciu uczenia nienadzorowanego. *LDA* reprezentuje dokumenty jako mieszanki tematów, które z kolei reprezentowane są jako rozkłady prawdopodobieństwa na zbiorze słów. Liczba tematów ustalana jest za pomocą hiperparametru. *LDA* jest modelem statystycznym, który wykorzystuje m.in. rozkład Dirichleta.

Rozkład Dirichleta to ciągły rozkład prawdopodobieństwa parametryzowany przez wektor α K dodatnich liczb rzeczywistych. Wymiar wektora α określa wymiar rozkładu. Gęstość rozkładu Dirichleta wyraża się wzorem 1.2:

$$f\left(x_1, \cdots, x_{K-1}; \alpha_1, \cdots, \alpha_K\right) = \frac{1}{\mathrm{B}(\alpha)} \prod_{i=1}^{K} x_i^{\alpha_i - 1}, \tag{1.2}$$

gdzie $x_1, \cdots, x_{K-1} > 0$, $x_1 + \cdots + x_{K-1} < 1$, $x_K = 1 - x_1 - \cdots - x_{K-1}$, a B to stała normalizująca. Nośnikiem gęstości rozkładu jest $K - 1$ wymiarowy sympleks.

W *LDA* rozkład Dirichleta jest wykorzystany w celu nadania początkowych wartości przynależności tematów do dokumentów oraz słów do tematów. Cechy rozkładu sprawiają, że tak

dobrane początkowe wartości parametrów modelu są zgodne z intuicją, że dokument pokrywa jedynie mały zestaw tematów, a temat zawiera najczęściej tylko mały zestaw słów. Wykorzystanie rozkładu Dirichleta do określenia wartości początkowych skutkuje w rezultacie lepszym dopasowaniem dokumentów i tematów.

Przypuśćmy, że mamy zestaw dokumentów. Wybieramy ustaloną liczbę T tematów, które zamierzamy wykryć. Chcemy użyć *LDA* w celu wyznaczenia reprezentacji każdego dokumentu jako mieszanki tematów oraz słów powiązanych z każdym tematem. Jednym ze sposobów, aby osiągnąć ten cel jest wnioskowanie oparte na próbkowaniu Gibbsa. Metoda ta działa zgodnie z następującymi krokami.

1. Przejdź przez każdy dokument i losowo (zgodnie z rozkładem Dirichleta) przypisz każde słowo dokumentu do jednego z T tematów. Warto zauważyć, iż etap ten daje pierwsze przybliżenie docelowej reprezentacji. W kolejnych krokach należy poprawiać to przybliżenie.

2. Dla każdego dokumentu d, dla każdego słowa w należącego do d, dla każdego tematu t oblicz: $p(t|d)$, czyli odsetek liczby słów w d, które są aktualnie przypisane do tematu t oraz oblicz $p(w|t)$, czyli odsetek liczby wystąpień słowa w, które są przypisane do tematu t w skali całego korpusu. Przypisz słowu w nowy temat poprzez losowanie z prawdopodobieństwem $p(t_i|d) * p(w|t)$ dla każdego tematu t_i.

Ciągłe wykonywanie powyższych kroków doprowadzi do stabilnej sytuacji, w której przestaną następować zmiany przypisań słów do tematów. Wtedy należy zakończyć działanie algorytmu.

Zaletą *LDA* jest jego interpretowalność: do każdego tematu przypisane są z pewną wagą prawdziwe słowa pochodzące z przetwarzanego korpusu. Metoda ta może być traktowana jako technika redukcji wymiarowości, gdyż dopasowuje dokumentowi składającemu się z wielu słów reprezentację złożoną z małej liczby tematów.

Osadzanie słów w przestrzeni wektorowej

Od 2013r., wraz z wprowadzeniem przez T. Mikolova metody *Word2vec* [19] nastąpił gwałtowny rozwój i niewątpliwy sukces metod typu *word embeddings*. Określenie *word embeddings* oznacza osadzanie słów w przestrzeni wektorowej przy pomocy uczenia nienadzorowanego i zostało po raz pierwszy użyte 2003r. w pracy Y. Bengio [2], gdzie wektory słów generowane są przez głęboką sieć neuronową. Ogół technik zaliczanych obecnie do *word embeddings* cechuje się dążeniem do reprezentacji słów wraz z zależnościami pomiędzy nimi w postaci wektorów o stosunkowo niskiej wymiarowości. Dzieje się to w opozycji do wcześniejszych podejść podobnych do *bag-of-words* — produkującego ogromne, rzadkie wektory, których wymiary równają się rozmiarowi

słownika, o który oparty jest model (rzędu setek tysięcy). Ważną własnością metod osadzania słów jest zachowanie przez wektory semantycznych i syntaktycznych właściwości słów, co pozwala wykonywać na nich operacje arytmetyczne odwzorowujące cechy tychże słów, np. *vector(„king")-vector(„man"+vector(„woman")* $\approx$ *vector(„queen")*

Stosowane obecnie podejścia generowania wektorowych reprezentacji słów można podzielić na dwa typy.

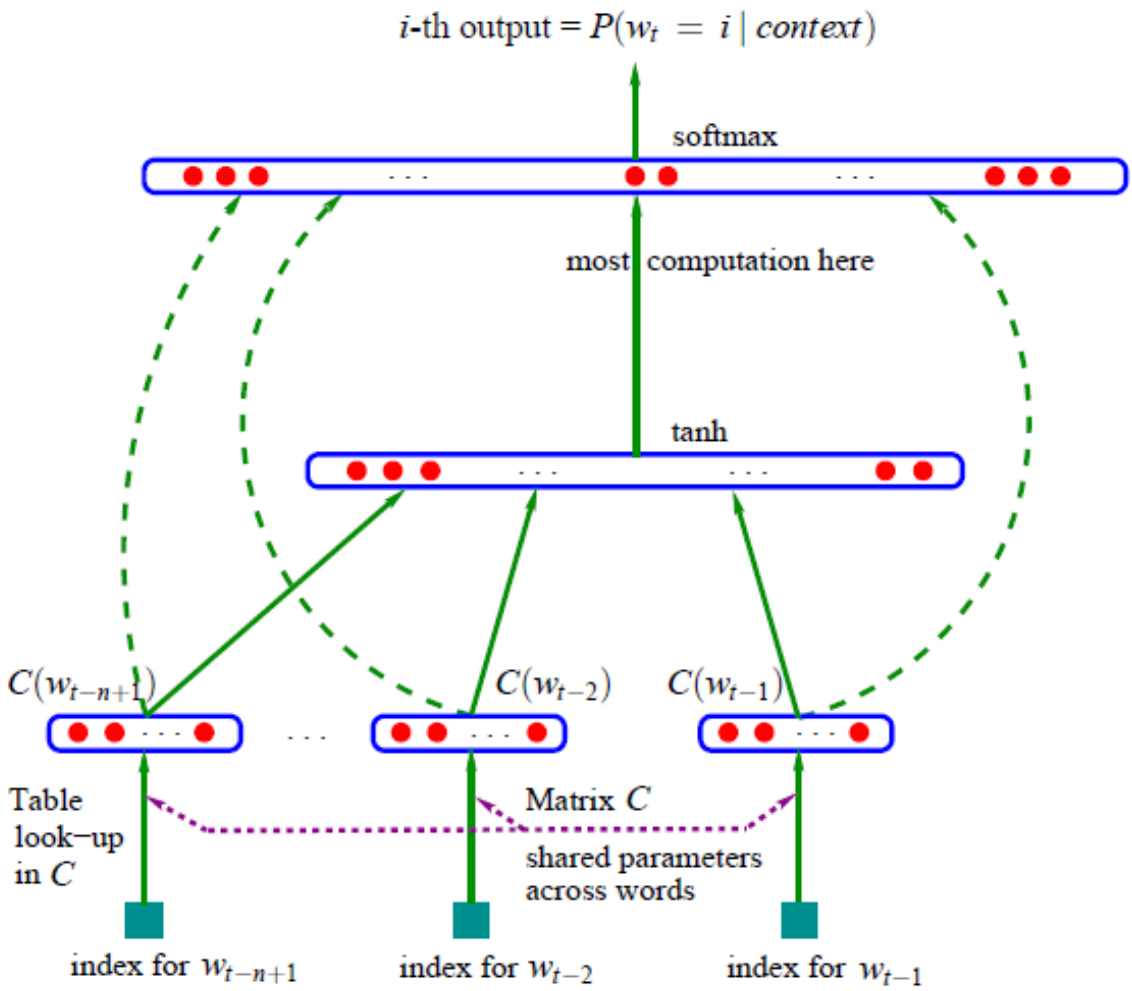
1. Modele predykcyjne: uczą się wektorowych reprezentacji słów poprzez zmniejszenie błędu predykcji słów należących do lokalnego kontekstu słowa i. Poniżej opisuję sztandarowy przykład takiego modelu — *Word2vec*, w którym sposobem na optymalizację funkcji celu jest zastosowanie płytkiej sieci neuronowej typu *feed-forward* optymalizowanej za pomocą metody *stochastic gradient descent*.

2. Metody oparte o zliczanie: generują wektory słów poprzez redukcję wymiarowości w globalnej macierzy współwystąpień słów. Jako pierwszy etap konstruują one ogromną (wymiar równa się liczbie słów w słowniku korpusu) macierz, która (podobnie, jak metodzie *LSI*) następnie ulega faktoryzacji, aby uzyskać macierz o mniejszym wymiarze, lecz nadal zachowującą powiązania pomiędzy słowami. Przykładem jest tu opisana poniżej metoda *Global Vectors — GloVe*.

Jedną z szerokiego wachlarza możliwości, jakie dają tego typu techniki jest określanie podobieństwa pomiędzy całymi dokumentami, wykorzystując dodatkowe metody pozwalające przenieść zależności między poszczególnymi słowami dokumentu na zależności między całymi zbiorami słów, co jest istotne z punktu widzenia tematu niniejszej pracy. Dwie z nich: metodę centroidu oraz *Word Mover's Distance* opisuję później w tym rozdziale. Warto zauważyć, iż posiadając wektorową reprezentację słów można wyznaczyć „odległość" pomiędzy dwoma dokumentami nawet, jeżeli nie posiadają one wspólnych słów.

Podejścia *deep learning*

Wspomniane podejście Bengio oparte jest o sieć neuronową typu f*eed-forward* o jednej warstwie ukrytej zgodnie z architekturą z poniższego rysunku.



Rysunek 1.1: Neuronowy model języka. Źródło: [2].

Celem działania sieci jest maksymalizacja funkcji celu postaci 1.3:

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \log f(w_t, w_{t-1}, \cdots, w_{t-n+1}), \tag{1.3}$$

gdzie $f(w_t, w_{t-1}, \cdots, w_{t-n+1})$ odpowiada prawdopodobieństwu $p(w_t \mid w_{t-1}, \cdots, w_{t-n+1})$ wystąpienia słowa w_t bezpośrednio po sekwencji słów $w_{t-1}, \cdots, w_{t-n+1}$. Wektorowa reprezentacja słowa uzyskiwana jest tu przez przemnożenie wejściowego wektora (wektor zer z jedynką na i-tym miejscu reprezentujący i-te słowo, *one-hot-vector*) z macierzą wag pierwszej warstwy sieci.

Podejście to jak i kolejne ([5]) wykorzystujące głębokie sieci neuronowe nie znalazły zastosowań komercyjnych, ponieważ ich wydajność nauki jest na tyle niska, że niemożliwe jest ich użycie przy ogromnych zbiorach danych wykorzystywanych w środowiskach produkcyjnych.

Rozwiązaniem tego problemu wydają się być nowe metody wektorowej reprezentacji słów powstałe na przestrzeni ostatnich lat. W odróżnieniu od metod opartych o *deep learning*, opierają

się one o metody szybkiej nauki, np. o płytkie sieci neuronowe, które uczą się na tyle krótko, że sprawdzają się one w zastosowaniach komercyjnych.

Word2vec

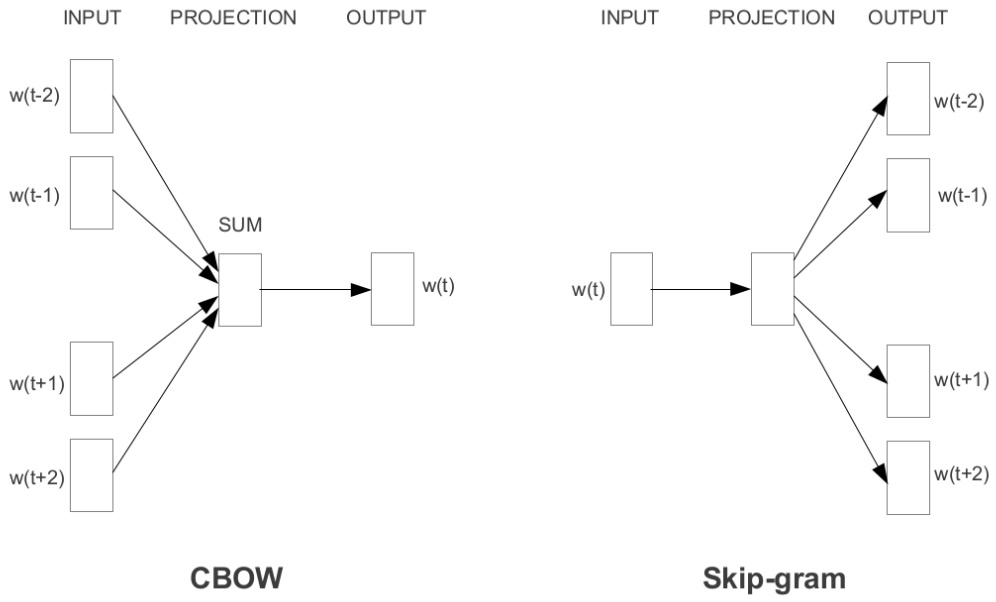
Metoda *Word2vec* wprowadzona w 2013r. w [19] odniosła niewątpliwy sukces w porównaniu z wcześniejszymi metodami osadzania słów w przestrzeni wektorowej. Autorzy metody proponują sieć neuronową, która podobnie, jak wcześniejsze podejścia ma za zadanie odtworzyć kontekst danego słowa i na tej podstawie dokonać reprezentacji słowa jako wektora liczb rzeczywistych. Różnica jest taka, iż sieć ta ani nie jest głęboka, ani też nie zawiera nieliniowych funkcji aktywacji wykorzystywanych we wcześniejszych modelach. Wyróżnia się dwie odwrotne architektury sieci:

- *CBOW* (*continuous bag-of-words*): na postawie okna N sąsiednich słów sieć przewiduje słowo, którego z największym prawdopodobieństwem te N słów jest sąsiedztwem. W tym modelu funkcja celu przyjmuje postać 1.4:

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \sum_{-n\leqslant j\leqslant n,\neq 0} \log p(w_{t+j} \mid w_t). \tag{1.4}$$

- *skip-gram*: na podstawie słowa sieć dokonuje predykcji N sąsiednich słów. Zadaniem sieci neuronowej jest wtedy optymalizacja funkcji celu postaci 1.5:

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \log p(w_t \mid w_{t-n}, \cdots, w_{t-1}, w_{t+1}, \cdots, w_{t+n}). \tag{1.5}$$



Rysunek 1.2: Schemat sieci wykorzystującej podejście *skip-gram* i *CBOW*. Źródło: [19].

Używając tej stosunkowo prostej architektury można przeprowadzić proces nauki używając milionów słów, których powiązania między sobą zostaną zachowane w systemie wag sieci neuronowej. Zaletą podejścia *skip-gram* są lepsze wyniki w przypadku rzadkich słów. Nauka w przypadku tego podejścia jest jednak wolniejsza niż dla *CBOW* [13].

W celu dalszego opisu metody *Word2vec* wprowadzam pojęcie funkcji *softmax* użytej w warstwie wyjściowej sieci neuronowej. *Softmax* jest generalizacją funkcji logistycznej, zamieniającą K-wymiarowy wektor z dowolnych liczb rzeczywistych na K-wymiarowy wektor liczb rzeczywistych z zakresu $[0, 1]$, które sumują się do 1. Funkcja wyraża się wzorem 1.6:

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \; dla \; j = 1, \; ..., K. \tag{1.6}$$

Wyjście funkcji można traktować jako pewien rozkład prawdopodobieństwa.

W metodzie *Word2vec* nauka polega na trenowaniu sieci neuronowej. Jednakże w odróżnieniu od innych metod wykorzystujących sieci neuronowe, *Word2vec* nie używa później wytrenowanej sieci jako takiej, a jedynie otrzymanych w wyniku nauki wag warstwy ukrytej sieci, które faktycznie są wynikowymi wektorami słów.

W dalszym opisie metody szczegółowo skupiam się na podejściu *CBOW*, lecz podejście *skip-gram* wygląda analogicznie.

Sieć neuronowa będąca wynikiem nauki przyjmuje na wejściu wektor binarny długości odpowiadającej liczbie słów w słowniku V zbudowanym na korpusie treningowym. Wektor ten wypełniony jest wartościami 0 oraz jedną wartością 1 na i-tej pozycji — *one-hot-vector*. Taki wektor odpowiada i-temu słowu ze słownika V. Wejściem sieci są kolejne słowa z korpusu w tej właśnie reprezentacji. Wyjściem sieci jest wektor tej samej długości o wartościach rzeczywistych z zakresu $[0, 1]$, w którym wartość na i-tej pozycji odpowiada prawdopodobieństwu, że i-te słowo ze słownika znajduje się w sąsiedztwie słowa wejściowego. Za „sąsiedztwo" wielkości x należy tu rozumieć zbiór złożony z x słów występujących przed danym słowem w korpusie i x słów położonych za danym słowem. Wartość x może być tu ograniczona przez początek/koniec zdania, które ograniczają kontekst danego słowa.

Jako efekt należy się spodziewać, że dla słowa wejściowego „Brytania" otrzymamy na wyjściu wysoką wartość prawdopodobieństwa dla słowa „Wielka", a niską np. dla słowa „skoroszyt".

Jednym z parametrów metody *Word2vec* jest wymiarowość przestrzeni, w której znajdują się otrzymane wektory odpowiadające słowom z korpusu. Liczba ta ma swoje źródło w wielkości warstwy ukrytej sieci neuronowej. Wagi warstwy ukrytej można interpretować jako macierz MxN, gdzie M to liczba słów słownika V (wielkość wektora wejściowego), a N to liczba neuronów w warstwie ukrytej. Po przeprowadzeniu nauki i-ty wiersz tej macierzy odpowiada wektorowi długości N, który reprezentuje i-te słowo ze słownika V.

W sieci nie jest używana funkcja aktywacji, ale prawdopodobieństwa na wyjściu są efektem działania funkcji *softmax*. Funkcja ta ma za zadanie sprowadzić wyjściowe wartości warstwy ukrytej do postaci rozkładu prawdopodobieństwa.

FastText

FastText [15] to biblioteka stworzona w 2016r. w celu wydajnego uczenia wektorowej reprezentacji słów oraz klasyfikacji zdań. Zasada działania metody bazuje na *Word2vec*. Od „klasycznego" *Word2vec* różni się jednak stopniem szczegółowości analizy słów. *Word2vec* traktuje słowo jaką najmniejszą, niepodzielną jednostkę, której wektorową reprezentację usiłuje wyznaczyć. *FastText* natomiast dokonuje analizy również wewnętrznej struktury słów. Wykorzystuje w tym celu rozbicie słowa na podsłowa — ciągi znaków o określonej długości n, (*character n-grams*). Np. słowo *pokój* składa się następujących 3-gramów: *pok*, *okó*, *kój*. Ostatecznie słowu wejściowemu zostaje przypisany wektor składający się ze średniej oryginalnej reprezentacji wektorowej słowa oraz wektorowych reprezentacji jego *n-gramów*.

Podejście takie daje szereg nowych możliwości. Pomaga wyznaczyć reprezentację wektorową rzadkich słów, które być może mają wspólny rdzeń (i znaczenie) z innymi, częściej występującymi słowami. Metoda pozwala również nadać wektorową reprezentację słowom, których w ogóle nie ma w słowniku, ponieważ ich podsłowa mogą należeć do słów w słowniku się znajdujących. Zalety te wydają się być szczególnie obiecujące w przypadku bogatych morfologicznie języków, np. języka polskiego, tureckiego, czy fińskiego.

GloVe

GloVe [24] (*GLObal VEctors*) jest kolejną wartą uwagi metodą *word embeddings* powstałą na przestrzeni ostatnich lat. Algorytm *GloVe* różni się od *Word2vec* w sposobie uzyskania wektorowej reprezentacji słów. *Word2vec* jest modelem predykcyjnym, natomiast trening w *GloVe* opiera się na globalnej macierzy współwystąpień słów. Ponadto, w porównaniu do *Word2vec*, *GloVe* stara się wyznaczyć reprezentacje wektorowe wprost, podczas gdy w *Word2vec* dzieje się „przy okazji" — szkoli się sieć neuronową nie w celu jej dalszego wykorzystania dla predykcji, a jedynie dla jej macierzy wag.

Algorytm *GloVe* składa się z następujących kroków [21]:

1. Zgromadź współwystąpienia słów w formie macierzy X. Każdy element X_{ij} takiej macierzy reprezentuje jak często słowo i występuje w pobliżu słowa j. Zazwyczaj macierz buduje się poprzez skanowanie bazowego korpusu oknem o ustalonej szerokości, w obrębie którego

centralne słowo leży w kontekście słów je otaczających. Dodatkowo można tu wprowadzić wagi dla słów malejące wraz ze wzrostem dystansu od słowa centralnego.

2. Zdefiniuj ograniczenie dla każdej pary słów: $w_i^T w_j + b_i + b_j = log(X_{ij})$, gdzie w_i oznacza wektor głównego słowa, w_j słowa leżącego w pobliżu i, b_i i b_j to skalary.

3. Zdefiniuj funkcję kosztu 1.7:

$$J = \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{ij})(w_i^T w_j + b_i + b_j - \log X_{ij})^2, \tag{1.7}$$

gdzie f jest funkcją ważącą, która pomaga zapobiec uczeniu tylko na podstawie najbardziej popularnych par słów. Autorzy proponują funkcję postaci 1.8:

$$f(X_{ij}) = \begin{cases} (\frac{X_{ij}}{x_{max}})^\alpha & \text{if } X_{ij} < XMAX \\ 1 & \text{w p.p.} \end{cases} \tag{1.8}$$

Celem funkcji optymalizacji funkcji kosztu jest minimalizacja różnicy pomiędzy iloczynami skalarnymi wektorów współwystępujących słów.

4. Dokonaj minimalizacji funkcji kosztu poprzez stopniową aktualizację wektorów w_i i w_j.

Odległość między wektorowymi reprezentacjami dokumentów

W celu wykorzystania omówionych metod osadzania słów, należy wybrać metodę obliczania odległości między całymi dokumentami, których słowa potrafimy reprezentować jako wektory. Zakładam, że jeżeli dystans pomiędzy dokumentami jest mały, to ich tematyka jest podobna.

Centroid

Najprostszą i najbardziej intuicyjną metodą obliczenia odległości pomiędzy wektorowymi reprezentacjami dokumentów jest wykonanie dwóch prostych kroków:

1. Uśrednienie wektorów wchodzących w skład każdego z dokumentów. Powstały w ten sposób wektor jest centroidem reprezentującym dokument w przestrzeni wektorowej.

2. Obliczenie dystansu między wektorami. Powszechnie przyjętą praktyką jest stosowanie tzw. odległości kosinusowej — znormalizowanego iloczynu skalarnego wektorów A i B. Jest to kosinus kąta pomiędzy dwoma wektorami reprezentującymi dokumenty. Zaletą tej metody jest natychmiastowa normalizacja wyniku do zakresu $[0, 1]$. Odległość wyraża się wzorem 1.9:

$$sim = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}, \tag{1.9}$$

gdzie A_i i B_i są składowymi wektorów A i B

Wadą opisanej powyżej metody jest utrata potencjalnie użytecznych zależności pomiędzy poszczególnymi wektorami wchodzącymi w skład dokumentu.

W kontrze do tego prezentuję metodę obliczania dystansu między dokumentami uwzględniającą rozkład wektorów wewnątrz dokumentu.

Word Mover's Distance

Word Mover's Distance (*WMD*) [17] to rozwiązanie zwracające odległość między dokumentami tekstowymi. W tym celu adaptuje algorytm *Earth Mover's Distance* (*EMD*) [29] oraz wektorową reprezentację słów dokumentu. *WMD* mierzy odległość między dokumentami jako minimalny dystans, jaki wektory słów pierwszego dokumentu muszą „pokonać", aby osiągnąć wartości wektorów z drugiego dokumentu.

EMD jest metodą mierzenia odległości pomiędzy dwoma rozkładami, która opiera się na minimalnym koszcie, jaki musi zostać poniesiony, aby dokonać transformacji jednego rozkładu w drugi. Problem można sformalizować jako problem programowania liniowego, gdzie: $P = \{f(p_1, w_{p_1})...(p_m, w_{p_m})\}$, $Q = \{f(q_1, w_{q_1})...(q_n, w_{q_n})\}$ są danymi rozkładami o m (odpowiednio n) klastrach p_i (q_j), a w_{p_i} (w_{q_j}) jest masą klastra. $D = [d_{ij}]$ jest macierzą odległości, w której d_{ij} reprezentuje odległość pomiędzy klastrami p_i i q_j. Celem jest znaleźć taki przepływ $F = [f_{ij}]$, gdzie f_{ij} to przepływ pomiędzy p_i i q_j, który minimalizuje całościowy koszt postaci 1.10

$$work(P, Q, F) = \sum_{i=1}^{m} \sum_{n}^{j=1} d_{ij} f_{ij} \tag{1.10}$$

przy odpowiednich ograniczeniach [29].

Przypuśćmy, że dzięki metodzie *Word2vec* dla słownika V o n słowach otrzymujemy macierz $X \in \mathbb{R}^{d \times n}$. i-ta kolumna tej macierzy reprezentuje i-te słowo ze słownika V. Odległości pomiędzy wektorami reprezentującymi semantycznie zbliżone słowa są relatywnie mniejsze od odległości dla słów niezwiązanych ze sobą. Celem *WMD* jest zawrzeć semantyczne podobieństwo pomiędzy poszczególnymi parami słów w dystansie pomiędzy całymi dokumentami. Aby to osiągnąć metoda traktuje dokument jako rozkład, którego i-tym elementem jest liczba wystąpień i-tego słowa w tym dokumencie, a następnie stosuje metodę *EMD* do obliczenia dystansu między tymi rozkładami. Macierz odległości D używana w metodzie *EMD* jest zbudowana na bazie odległości między wektorami *Word2vec* reprezentującymi słowa dokumentów. $d_{ij} = ||x_i - x_j||$, gdzie i i j to indeksy słów ze słownika V, a x_{ij} to element macierzy X. *EMD* jest dobrze zbadanym problemem transportowym [29], a efektywne metody jego rozwiązania przy złożoności poniżej $O(p^3 \log p)$ zostały opisane np. w [23].

Miary oceny wyszukiwania

Razem z rozwojem systemów wyszukiwania informacji powstały miary pozwalające ocenić wyniki działania tych systemów. Do najbardziej popularnych należą *precision* (precyzja) i *recall* (zwrot).

Precision to odsetek wyszukanych dokumentów (*retrieved*), które są relewantne (*relevant*) do zapytania (1.11):

$$\text{precision} = \frac{|\{\text{relevant}\} \cap \{\text{retrieved}\}|}{|\{\text{retrieved}\}|}. \tag{1.11}$$

Recall natomiast jest liczbą wyszukanych relewantnych dokumentów w stosunku do wszystkich relewantnych dokumentów (również tych niewyszukanych) (1.12).

$$\text{recall} = \frac{|\{\text{relevant}\} \cap \{\text{retrieved}\}|}{|\{\text{relevant}\}|}. \tag{1.12}$$

Bardziej złożoną miarą lepiej charakteryzującą jakość procesu wyszukiwania jest *F-miara*. W celu obliczenia wyniku uwzględnia ona zarówno precyzję jak i zwrot. Miara wyraża się wzorem 1.13:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}} \tag{1.13}$$

Rezultat miary można interpretować jako ważoną średnią precyzji i zwrotu.

Powyższe miary dokonują binarnego podziału wyników wyszukiwania: na relewantne i nierelewantne. Jednakże część rekomendowanych przedmiotów może być mniej lub bardziej od innych istotna dla użytkownika. W tym świetle listę rekomendacji można uznać za pewien ranking, którego elementy mogą być uszeregowane lepiej lub gorzej. Ocena, czy dane uszeregowanie jest dobre, jest istotnym problemem z punktu widzenia ewaluacji metod generujących elementy rankingu. Stąd zachodzi potrzeba wprowadzenia formalnej miary jakości uszeregowania elementów rankingu. Można osiągnąć poprzez rozszerzenie podstawowych metod ewaluacji opartych na binarnej ocenie relewantności, jak *recall* i *precision*.

Miara *nDCG* [14] (*Normalized Discounted Cumulative Gain*) jest miarą jakości rankingu, która opiera się na założeniu, że im bardziej relewantne wyniki, tym wyżej powinny być w rankingu, aby ranking był najbardziej wartościowy. Miara ta mierzy skumulowany „zysk" powstały poprzez umieszczenie poszczególnych przedmiotów na określonych pozycjach rankingu. Miara *nDCG* jest znormalizowaną wersją miary *DCG* (*Discounted Cumulative Gain*), która wyraża się wzorem 1.14:

$$\text{DCG}_\text{p} = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)}, \tag{1.14}$$

gdzie p to liczba elementów rankingu, i to miejsce przedmiotu w rankingu, a rel to poziom relewantności elementu. DCG premiuje relewantne przedmioty, które są wysoko w rankingu oraz karze za relewantne przedmioty w dole rankingu. W wariancie $nDCG$ następuje jeszcze normalizacja przez podzielenie wartości DCG rzeczywistego rankingu przez DCG idealnego rankingu ($IDCG$, 1.15) zbudowanego na elementach korpusu ułożonych malejąco pod kątem relewantności.

$$\mathrm{IDCG_p} = \sum_{i=1}^{|REL|} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{1.15}$$

REL oznacza listę relewantnych przedmiotów z całego zbioru podlegającego szeregowaniu. Miara $nDCG$ wyraża się wzorem 1.16:

$$\mathrm{nDCG_p} = \frac{DCG_p}{IDCG_p}. \tag{1.16}$$

Dane

Dane, na których testowane są opisywane w niniejszej pracy metody otrzymałem dzięki życzliwości serwisu *Allegro*, podpisując umowę o poufności. Stąd, w niniejszej pracy brak jakichkolwiek przykładów danych poza tymi dostępnymi publicznie za pośrednictwem strony internetowej *Allegro*. W niniejszym rozdziale opisuję strukturę i stan pozyskanych danych oraz proces ich wstępnego przetwarzania i analizy w celu zastosowania na nich zaadaptowanych metod opisanych w rozdziale poprzednim.

Opis danych

Otrzymane dane to baza ok. 20000 artykułów tekstowych w formacie *JSON*. Są to te same artykuły, które są dostępne dla użytkowników poprzez serwis internetowy (stan na styczeń 2017). Pojedynczy rekord danych składa się z głównej treści artykułu oraz z metadanych, z których za istotne z punktu widzenia tematu pracy uznałem pola: *id*, *kategoria* i *słowa kluczowe*.

Treść artykułu

Treść każdego artykułu składa się z trzech pól: *zawartość*, *nagłówek* i *tytuł*. Średnia długość „surowego" artykułu to 821 słów, w tym nagłówek to jednozdaniowy wstęp. Średnią tę estymuję na podstawie średniej liczby znaków artykułu i średniej długości słowa w języku polskim. Dokładne statystyki tekstu będą dostępne dopiero po wstępnym przetwarzaniu.

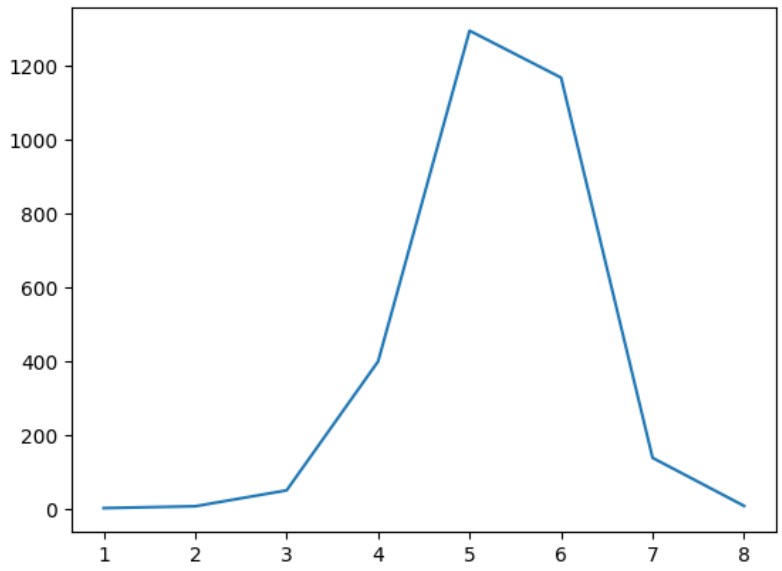
Wszystkie artykuły napisane są w języku polskim, w nielicznych przypadkach wykryłem błędy, tzw. „literówki". Jako, że artykuły ze zbioru dotyczą produktów sprzedawanych za pośrednictwem serwisu *Allegro*, w skład słownika zbudowanego na ich bazie wchodzi wiele słów specyficznych dla różnych branż. Są to m.in. nazwy modeli aparatów (np. „Sony Alpha 77 II"), samochodów, gier komputerowych, a także nazwy techniczne: „sprężarka", „hipertoniczny", „autofocus".

Artykuły posiadają w swej treści wiele znaczników interpretowanych przez system, na podstawie których wzbogacana jest warstwa wizualna strony internetowej zawierającej artykuł, np. obrazki czy łącza do ofert związanych z tematem artykułu.

Spójność danych oceniam na wysoką, tj. każde pole zawarte w strukturze dokumentu jest zawsze wypełnione — brak jest wartości typu *NULL*.

Kategoria

Każdy artykuł został przez autora przydzielony do pewnej kategorii, która odpowiada tematyce artykułu, np. „Aparaty cyfrowe" czy „Przyprawy i zioła". W skład pola *kategoria* wchodzi również lista kategorii nadrzędnych, a cała hierarchia kategorii ma strukturę drzewiastą. Np. kategoria nadrzędna dla kategorii „Przyprawy i zioła" to „Delikatesy", a dla kategorii „Delikatesy" to „Dom i zdrowie". Każdy artykuł należy do tylko jednej kategorii będącej dowolnym węzłem w drzewie (nie tylko liściem). Drzewo kategorii posiada 8 poziomów, a najwięcej węzłów znajduje się na poziomach 5 i 6. Najwięcej artykułów jest przypisanych do kategorii z poziomów 4 i 5, średnio artykuł jest przypisany do kategorii z poziomu 4,63.



Rysunek 2.1: Wykres przedstawia liczby kategorii na poszczególnych poziomach drzewa kategorii.

Po wstępnej analizie system kategorii oceniam jako spójny i rzetelny. Uważam, że można użyć go jako punkt odniesienia przy konstrukcji metod ewaluacji testowanych technik określania podobieństwa między artykułami.

Słowa kluczowe

Do każdego artykułu dołączona jest lista słów kluczowych. Są to wyrażenia złożone z jednego lub kilku słów, które mają za zadanie scharakteryzować w skrócie jego zawartość, np. „aparaty",

„aparaty cyfrowe", „lustrzanki", „Sony". Pole to jest wykorzystywane w dotychczasowym mechanizmie generowania rekomendacji - artykuły podobne do danego są wyszukiwane na podstawie jego słów kluczowych. Przyglądając się wszystkim słowom kluczowym zestawionym razem zauważyłem pewne niespójności: część słów kluczowych o tej samej treści pisana jest w inny sposób, np. różną wielkością liter, czy używając myślnika zamiast spacji. Wynika to zapewne z faktu przypisywania słów kluczowych samodzielnie przez autorów w oderwaniu od słów przypisanych do reszty artykułów. Średnio każdy artykuł ma przypisane 5,8 słów kluczowych, natomiast unikalnych słów kluczowych w skali całego korpusu jest 60403.

Wstępne przetwarzanie danych

W celu zwiększenia skuteczności metod analizy tekstu stosuje się wstępne przetwarzanie danych. Ma ono na celu takie przygotowanie tekstu, aby zmaksymalizować jakość wyników operujących na nim później algorytmów. Techniki wstępnego przetwarzania tekstu nie wchodzą w skład żadnego standardu — dobieram je indywidualnie do konkretnego przypadku, zgodnie z intuicją.

Niżej opisuję kolejne kroki wstępnego przetwarzania tekstu, które wykonuję na posiadanym zbiorze artykułów.

1. Oczyszczanie tekstu ze zbędnych, wspomnianych wcześniej znaczników. Z punktu widzenia semantycznej analizy tekstu są one bezużyteczne, czy wręcz szkodliwe (powodują pewne „zanieczyszczenie" tekstu). Stąd usuwam je wykorzystując odpowiednio skonstruowane wyrażenia regularne. Przykładem takiego znacznika jest `![2_new.jpg](http://(...)'2_new.jpg')` umieszczający obrazek w środku tekstu (treść adresu *URL* usunąłem przyczyn poufności).

2. Usunięcie słów stopu (*stopwords*) — na ogół krótkich słów niewnoszących nic do znaczenia całości artykułu. Są to np. „w", „z", „ponieważ". Ich usunięcie zmniejsza liczbę słów dokumentu skracając tym samym czas jego przetwarzania. Jako, że słowa te występują często, usunięcie ich daje możliwość uwypuklenia znaczenia innych słów mających wpływ na rzeczywiste znaczenie całego artykułu. Zbiór słów stopu czerpię z [31].

3. Sprowadzenie wszystkich słów dokumentu do małych liter. Pomaga to ujednolicić postać części słów o tym samym znaczeniu, wśród których jedno występuje na początku zdania, a inne w środku.

4. Rozbicie słów połączonych myślnikiem. Doświadczenie w późniejszym etapie (tokenizacji) pokazuje, że narzędzie jej dokonujące (*Morfologik* [20]) nie radzi sobie z tego typu słowami

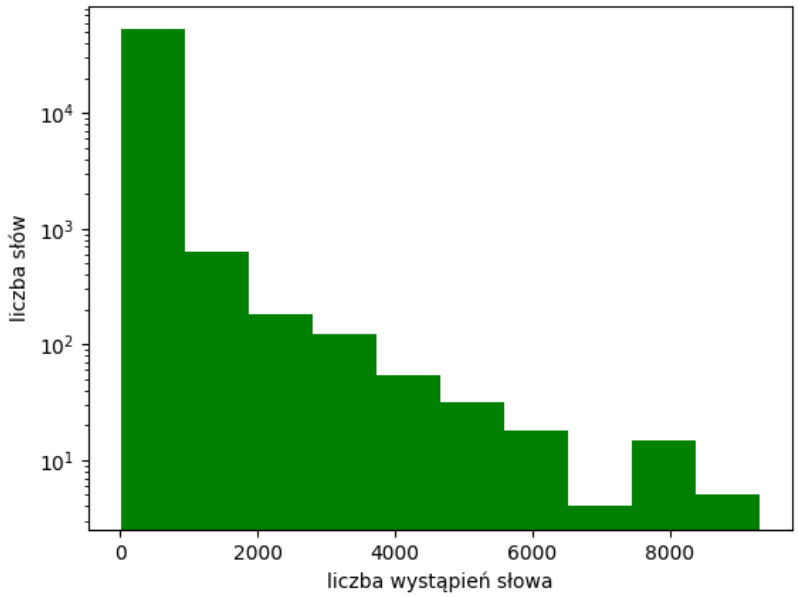
i zostawia je w niezmienionej postaci gramatycznej (np. „biało-czerwonego"). Stąd konieczność ręcznego wykonania przeze mnie mechanizmu rozbijającego takie słowa do postaci kompatybilnej z tokenizerem. Do wykonania odpowiedniej funkcji potrzebna była wcześniejsza analiza tego typu słów pod kątem zachowania obu członów w zależności od ich rodzaju, przypadku i występowania konkretnych liter w sufiksach słów składowych. Zależało mi także, aby nie rozbijać słów będących nazwami własnymi, czy symbolami urządzeń.

5. Tokenizacja. Jest to najistotniejszy element całego procesu. Polega na sprowadzaniu słów o tym samym znaczeniu, a różnej formie gramatycznej do tej samej postaci. Sporym utrudnieniem jest tutaj stopień skomplikowania języka polskiego oraz liczba wyjątków, jaką ten język posiada. Za przykład może posłużyć słowo „mieć", którego jedna z form to „ma", kolejna to „miej". Celem etapu jest sprowadzenie każdego z tych wyrazów do formy podstawowej „mieć". Do przeprowadzenia tej operacji stosuję narzędzie *Morfologik* [20].

Użycie wymienionych technik nie jest jedynym standardem, a wynikiem analizy przetwarzanych danych i techniki te zostały dobrane dla tego konkretnego przypadku

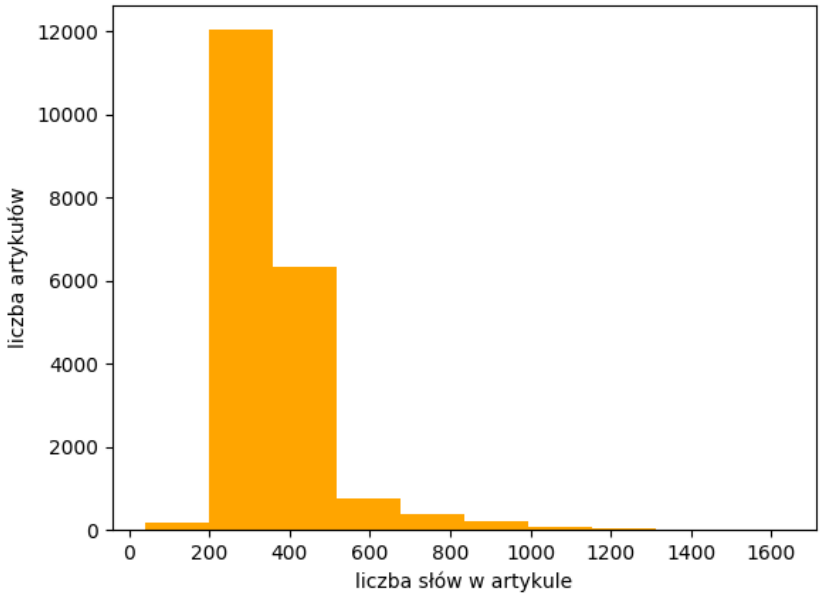
Opis danych po wstępnym przetwarzaniu

Powyższe kroki doprowadzają dane do stanu, w którym można zastosować techniki semantycznej analizy tekstu. Słownik zbudowany na wstępnie przetworzonym korpusie zawiera 98174 unikalnych słów, oraz 7409145 wszystkich słów (z powtórzeniami). Poniższy histogram (2.3) pokazuje, że przytłaczającą większość słów słownika zbudowanego na korpusie stanowią słowa występujące rzadko. Ciekawym wydaje się być fakt dużej liczby słów słownikowych występujących często (ok. 8000 razy) w stosunku do słów występujących rzadziej (ok. 7000 razy) (rysunek 2.2). Przykładem najczęściej występujących słów są: „sam", „uwaga", „ważny", „należeć", „wybrać", „sprawdzić", „model", „miejsce", „znaleźć". Najrzadziej występujące słowa to: „naciągactwo", „phone'ów", „v90", „eurobusiness", „namakać", „bale'a", „hmb", „ameksyka", „e-paper", „süskind". Widać wśród nich słowa bardzo nietypowe, również w języku codziennym, symbole modeli i marek oraz błędy, tzw. „literówki".



Rysunek 2.2: Histogram liczby wystąpień słów w korpusie w skali logarytmicznej.

Większość artykułów okazała się być podobnej długości. Średnia długość artykułu to 370 słów, co potwierdza rysunek 2.3.



Rysunek 2.3: Histogram długości artykułów.

Metody ewaluacji

W celu porównania stosowanych metod wyznaczania podobieństwa między artykułami konieczna jest formalizacja pewnych miar tego podobieństwa. W opisie znanych ogólnych miar posłużyłem się pojęciem „relewantności" — formalną wartością wyrażoną za pomocą liczb rzeczywistych. W praktyce rzadko jednak dysponuje się wartością, na ile dany element rankingu jest adekwatny do zapytania generującego ów ranking.

Ewaluacja rankingu, w którym trafność wyników zależy od zachowania realnych użytkowników jest zadaniem nietrywialnym. Podobieństwo artykułów napisanych w języku naturalnym jest rzeczą subiektywną. W sytuacji idealnej dysponowałbym obiektywną miarą podobieństwa pomiędzy parami N artykułów (np. wyznaczoną wcześniej przez miarodajną grupę użytkowników), które to N artykułów stanowiłoby zbiór testowy. Uzyskanie takich danych wiąże się jednak z dużymi kosztami i leży poza moimi możliwościami.

Inną praktyką umożliwiającą obiektywną ocenę, wykorzystywaną w działających systemach są tzw. *testy A/B* polegające na podziale użytkowników na grupy i zaaplikowaniu każdej grupie innego rozwiązania. Następnie mierzone są pewne wskaźniki wśród każdej grupy (w moim przypadku np. liczba „kliknięć" prawdziwych użytkowników w artykuły rekomendowane) i spośród zgromadzonych wyników wybierane jest rozwiązanie najlepsze.

Z powodu braku możliwości wykorzystania do ewaluacji realnych użytkowników serwisu internetowego jestem zmuszony wprowadzić własne miary oparte na dostępnych danych. Należy tu zaznaczyć niedoskonałość wprowadzanych miar, ponieważ każda z nich opiera się na pewnych założeniach, od których prawdziwości zależy jakość całej miary.

Testowane metody adaptuję tak, aby na podstawie pewnego artykułu bazowego otrzymywać listę artykułów podobnych do niego, uszeregowanych pod kątem relewantności malejąco. Takie działanie można sformalizować w postaci funkcji 3.1:

$$S_p : a_j \rightarrow \{a_i\}_{i<p}, \tag{3.1}$$

gdzie a to artykuł, a p to liczba elementów zwracanego ciągu. Funkcja S przyjmuje artykuł tekstowy i zwraca skończony ciąg artykułów do niego podobnych zgodnie ze stopniem dopasowania (najlepsze na początku). Celem działania niżej opisanych miar jest każdej parze postaci:

artykuł wejściowy oraz jeden z wyznaczonych artykułów podobnych przypisać ocenę tego podobieństwa — relewantność. Np. metoda S dla artykułu X zwraca ciąg Y_1, Y_2, ... Dla każdej z par za pomocą poniższych metod ewaluacji można określić relewantność np. $relevance(X, Y_1) = a_1, relevance(X, Y_2) = a_2$ itd. Następnie wyniki dla metody S i wejścia X należy zagregować. Dokonuję tego na dwa sposoby.

- Obliczenie średniej $\frac{1}{p}\sum_{i=1}^{p} relevance(X, Y_i)$.
- Użycie metody *nDCG*, gdzie relewantność to wynik poszczególnych ewaluacji podobieństwa artykułów.

Miara 1: Dystans oparty na metadanych

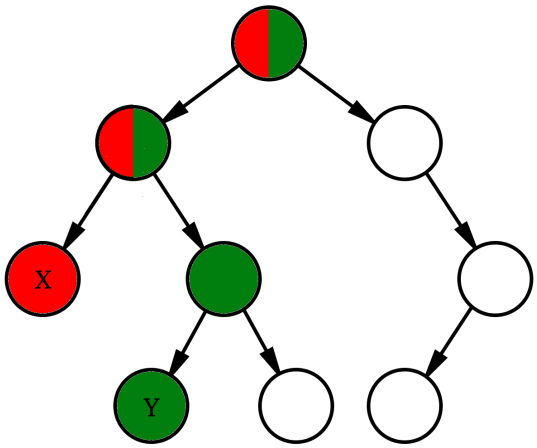
Jak wspomniałem wcześniej, dane prócz treści artykułów zawierają również pewne metadane, a wśród nich umożliwiające tworzenie powiązań między artykułami. Skupiam się tu na polach: *słowa kluczowe* i *kategoria*.

Kategorie

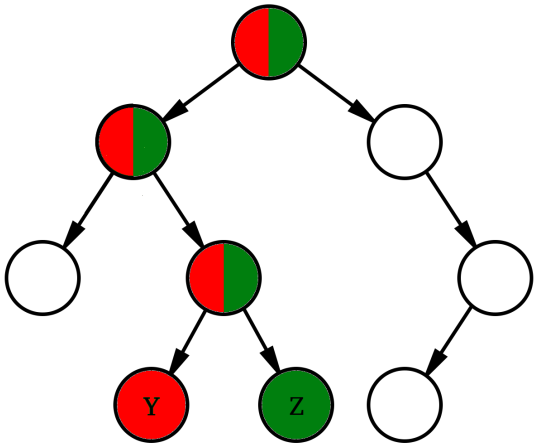
Pierwszą zastosowaną miarą, pozwalającą ocenić jakość dopasowania podobnych artykułów jest ich odległość we wcześniej wspomnianym drzewie kategorii. Formalnie wartość miary to długość części wspólnej ścieżek od korzenia drzewa kategorii do węzłów reprezentujących artykuły. Im więcej wspólnych przodków w drzewie, tym bardziej podobne do siebie są artykuły reprezentowane przez węzły drzewa. Zaletą miary jest fakt, iż przypisanie artykułu do kategorii zostało wykonane przez autora, którego można określić ekspertem w dziedzinie tematyki artykułu. Stąd przynależność artykułu do danej kategorii jest mocno uzasadniona. Kolejną zaletą tej miary jest fakt, iż można ją zastosować automatycznie — wiedza ekspercka jest już zapisana w metadanych artykułów. Należy zaznaczyć tu jednak, że miara nie jest idealna — każdy artykuł należy do tylko jednego liścia drzewa kategorii. Stąd artykuł poruszający zagadnienia z różnych obszarów, który można by przypisać dwóm stosunkowo odległym kategoriom A i B, zostanie przypisany tylko do jednej kategorii, np. A. Miara pokaże wtedy dużą odległość od artykułów z kategorii B, co nie jest prawdą. Wartości miary normalizuję, dzieląc je przez głębokość drzewa kategorii: 8 — wartość tę uzyskałem przy wcześniejszej analizie metadanych artykułów.

Za przykład mogą posłużyć drzewa na poniższych rysunkach. W drzewie 3.1. artykuły X i Y mają dwie wspólne kategorie, stąd $relevance(X, Y) = \frac{2}{8} = 0.25$. Natomiast artykuły Y i Z

z drugiego drzewa (3.2) mają trzech wspólnych przodków, stąd $relevance(Y, Z) = \frac{3}{8} = 0.375$. Miara wskazuje, że artykuły X i Y są do siebie mniej podobne, niż artykuły Y i Z.



Rysunek 3.1: Drzewo kategorii dla przykładu 1.



Rysunek 3.2: Drzewo kategorii dla przykładu 2.

Słowa kluczowe

Kolejna miara oparta na metadanych artykułów korzysta ze słów kluczowych. Wśród ogółu słów kluczowych występują niespójności, których większość zlikwidowałem poprzez sprowadzenie słów od małych liter oraz usunięciu słów stopu. Po tej unifikacji liczba unikalnych słów kluczowych to 58565.

Niniejsza miara działa podobnie do poprzedniej opartej o kategorie. Para artykułów otrzymuje 1 punkt za każdą wspólną parę posiadanych słów kluczowych. Następnie wartość jest normalizowana poprzez podzielenie przez maksymalną liczbę wspólnych słów kluczowych wywnioskowaną przy analizie metadanych artykułów, tj. 6. Np. dla artykułu X o słowach kluczowych {„kosmetyki", „kremy", „zmarszczki"} oraz Y o słowach kluczowych {„kosmetyki", „żele do włosów",

„kremy", „szampony"} $relevance(X, Y) = \frac{2}{6} = 0.(3)$, ponieważ oba artykuły mają dwa wspólne słowa kluczowe („kosmetyki" i „kremy").

Miara 2: Historyczna aktywność użytkowników serwisu

Gromadzenie informacji o aktywności użytkownika w ramach serwisu internetowego jest powszechną praktyką. Proces ten pozwala na analizę zachowania użytkowników, co może doprowadzić do wniosków, jakie usprawnienia należy przedsięwziąć, aby spełnić cele biznesowe. Jednym z przykładów aktywności użytkownika zapisywanej przez serwis *Allegro* są kliknięcia w linki znajdujące się na stronie internetowej. Informacja ta pozwala sporządzić jeszcze jedną miarę jakości dopasowania podobnych do siebie artykułów. Postać danych, jakie udało mi się uzyskać z serwisu to tabela o polach: adres strony, na której nastąpiło kliknięcie, adres strony, na którą prowadzi link, data kliknięcia.

Jak już zostało opisane powyżej dostępna w serwisie *Allegro* strona z artykułem tekstowym zawiera listę $(y_1,\ y_2,\ \ldots\ , y_p)$ odnośników do innych artykułów poruszających tematykę podobną do niego. Skoro zapisywana jest informacja o przejściach pomiędzy podstronami serwisu, to można obliczyć ile razy w pewnym okresie użytkownicy dokonali przejścia z artykułu x na rekomendowany do niego artykuł y_i, a ile razy na rekomendowany artykuł y_j $(i, j \leqslant p, i \neq j)$. Jeżeli liczba takich przejść z artykułu x na artykuł y_i jest większa niż z artykułu x na y_j, można wnioskować, iż y_i jest bardziej relewantną rekomendacją dla artykułu x niż y_j. Ostatecznie liczbę przejść z artykułu x na rekomendowany do niego artykuł y przyjmuję jako miarę relewantności między x i y wykorzystywaną w metodzie *nDCG*.

Przyjmijmy, że testowana metoda wyznaczania rekomendacji zwraca pewien ciąg artykułów $(z_1,\ z_2,\ \ldots,\ z_p)$ podobnych do danego artykułu x w kolejności od najbardziej relewantnego. Zadaniem niniejszej metody ewaluacji jest ocenić jakość tego uszeregowania, przyjmując za punkt odniesienia relewantność wg. powyższej definicji. Zgodnie z nią relewantność do artykułu x jest określona jedynie dla p artykułów zarekomendowanych do niego wcześniej przez serwis *Allegro* (tych, na które przejść mogli dokonywać użytkownicy). Stąd, definicja nie określa relewantności artykułu z_i, jeżeli ten nie znajduje się w ciągu $(y_1,\ y_2,\ \ldots\ , y_p)$. Dlatego, aby wykorzystać tu metodę *nDCG* należy wziąć pod uwagę jedynie elementy obu ciągów (y_i) i (z_i) należące do przecięcia zbiorów: $\{y_i\} \cap \{z_i\} = W$ — wszystkie artykuły należące do takiego zbioru mają określoną relewantność. Następnie przyjmuję uszeregowanie $(w_1,\ w_2,\ \ldots,\ w_{|W|})$ takie, że $i < j \Rightarrow relevence(x, w_i) > relevance(x, w_j)$ za idealne uszeregowanie z definicji *nDCG*. Natomiast za oceniany ranking przyjmuję ciąg złożony z elementów ciągu (z_i) takich, że $z_i \in W$.

Opis metody uzupełniam przykładem. Dotychczasowa metoda *Allegro* artykułowi x przypisała artykuły podobne $A = (a,\ b,\ c,\ d,\ e)$. Na podstawie historycznej aktywności użytkowników można określić, że $relevance(x, a) = 50$, $relevance(x, b) = 10$, $relevance(x, c) = 15$, $relevance(x, d) = 0$, $relevance(x, e) = 30$. Testowana metoda wyznaczyła dla artykułu x rekomendacje w postaci listy: $B = (c,\ f,\ a,\ b,\ d)$. Przecięciem zbiorów złożonych z elementów ciągów A i B jest $\{a,\ b,\ c,\ d\}$. Stąd, za idealne uszeregowanie przyjmuję ciąg $(a,\ c,\ b,\ d)$, natomiast za testowany ranking przyjmuję ciąg $(c,\ a,\ b,\ d)$. Oba te ciągi oraz funkcję *relevance* traktuję jako wejście dla metody *nDCG*, której wynikiem w tym przypadku jest wartość ok. 0.63. Przyjmuję ją jako wynik niniejszej metody ewaluacji dla metody generującej rekomendacje do artykułu x postaci $(c,\ f,\ a,\ b,\ d)$.

Zaletą metody jest, iż można ją zastosować automatycznie, lecz jest zależna od danych analitycznych pochodzących z serwisu, które są niedoskonałe. Istotną wadą jest fakt, że w wielu przypadkach przecięcie W jest puste lub zawiera mało elementów, co działa na niekorzyść jakości tej metody.

Miara 3: Ocena przez użytkowników offline

Ostatnią opracowaną przeze mnie miarą jest subiektywna ocena ekspercka. W celu obiektywizacji oceny, ewaluacja powinna być dokonana przez reprezentatywną grupę T osób operujących na tych samych danych. W metodzie tej grupa użytkowników dokonuje oceny podobieństwa par artykułów generowanych przez poszczególne testowane metody generowania rekomendacji.

1. Wybieram n losowych artykułów bazowych $b_j, j = 1, ..., n$.
2. Testowaną metodą M generuję p artykułów $s_{jk}, k = 1, ..., p$ podobnych do każdego b_j z n wcześniej wylosowanych.
3. Grupuję artykuły w pary: artykuł bazowy b_j — artykuł podobny s_{jk}.
4. Za pomocą stworzonego interfejsu prezentuję każdemu z użytkowników kolejno wygenerowane pary w postaci rzeczywistych stron z artykułami dostępnych przez przeglądarkę internetową. Co ważne każdy użytkownik otrzymuje ten sam zestaw par. Podczas prezentacji pary artykułów użytkownik za pomocą interfejsu webowego ocenia podobieństwo pomiędzy artykułami $rel(b_j, s_{jk})$ w skali 1-10.
5. Dla każdej pary obliczam średnią ocen wszystkich T użytkowników wyrażoną wzorem 3.2:

$$avg(j, k) = \frac{1}{T} \sum_{t}^{T} rel_t(b_j, s_{jk}), \tag{3.2}$$

gdzie rel_t to ocena t-go użytkownika.

6. Dla każdego artykułu bazowego b_j obliczam 3.3:

$$m(j) = \frac{1}{\sum_{i=1}^{p} i} \sum_{i=0}^{k} avg(j,k) * (p-i) \tag{3.3}$$

będące średnią ważoną ocen, gdzie każdy kolejny artykuł podobny otrzymuje coraz mniejszą wagę.

7. Oceną testowanej metody jest średnia $\frac{1}{n} \sum_j m(j)$.

Wadą tej metody jest jej powolność i potrzeba zaangażowania dodatkowych osób dokonujących ewaluacji. Niemożliwym wydaje się przeprowadzenie badania dla wszystkich artykułów, stąd konieczny jest wybór losowej próby artykułów, które parami poddane zostaną ocenie pod kątem podobieństwa.

Opis testów

W przeprowadzanych testach staram się dokonać porównania pomiędzy różnymi konfiguracjami tej samej metody oraz pomiędzy najlepszymi wariantami różnych metod. W tym celu wykorzystuję wprowadzone miary ewaluacji. W większości przypadków używam miar automatycznych, bazujących na metadanych artykułów oraz na historycznej aktywności użytkowników serwisu. Do ostatecznego porównania pomiędzy najlepszymi wariantami testowanych metod stosuję również miarę opartą na ocenie eksperckiej. Miara ta daje najbardziej rzetelne wyniki, jednakże jej użycie jest wyjątkowo kosztowne — wymaga zaangażowania osób testujących oraz sporych nakładów czasowych. Stąd zdecydowałem na użycie jej tylko w jednym przypadku.

Testowane metody generowania rekomendacji

Wykonuję szereg testów adaptacji metod semantycznej analizy języka naturalnego o różnych nazwach, w różnych konfiguracjach. Stąd dla czytelności wprowadzam niekiedy w nawiasach skrócone sygnatury tych metod, które stosuję w dalszej części pracy zamiast pełnych opisów.

Metody oparte o modelowanie tematu

Najważniejszym hiperparametrem metod tej grupy: *LSI* oraz *LDA* jest liczba tematów, stąd dokonuję testu, jak zmiana tego parametru wpływa na jakość modelu.

Metody oparte o *word embeddings*

Wykonuję testy adaptacji metod wektorowej reprezentacji słów dla wielu konfiguracji. Poniżej obszary, na których dokonuję zmian konfiguracji.

Korpus bazowy

Pierwszy model *Word2vec*, z jakim miałem styczność to model [16] stworzony m.in. przez dr inż. M. Piaseckiego z Politechniki Wrocławskiej dostępny poprzez stronę internetową. Model

ten był uczony na korpusie *Słowosieci* wer. 10 [34]. Zgodnie z opisem autorów dane przed uczeniem przeszły segmentację, lematyzację i ujednoznacznianie morfosyntaktyczne. Użyte parametry uczenia *Word2Vec*: metoda *skip-gram*, wektory długości 100, okno kontekstu wielkości 5.

Model ten zawiera 73875 spośród 98174 (75%) unikalnych słów korpusu artykułów *Allegro* oraz 7313915 z 7409145 (99%) wszystkich słów korpusu artykułów. Wskazuje to, iż słowa nieobecne w modelu są bardzo mało popularne w korpusie artykułów *Allegro* (stanowią ok. 1% całości). Po samodzielnym sprawdzeniu stwierdzam, że słowa nieobecne w modelu to: „literówki" lub słowa niepoprawnie stokenizowane (np. „urządzeia"), symbole marek produktów (np. „ux305fa", „i7-4700qm"), żargon branżowy (np. „bootsów"), złożenia wyrazów (np. „kurzoodporne"), wyrazy obce lub ich spolszczenia (np. „thermoprotect"). Uważam, iż mimo niewielkiej liczby tych słów w stosunku do wielkości korpusu mogą mieć one znaczący wpływ na semantykę artykułów.

W związku z powyższym stwierdzeniem wykonuję naukę modelu *Word2vec* również na samym korpusie artykułów *Allegro* po to, aby posiadać model zawierający wszystkie słowa występujące w artykułach. Najrozsądniejszym postępowaniem byłoby tutaj rozszerzenie modelu opartego na korpusie *Słowosieci* również o brakujące słowa, jednak metoda *Word2vec* nie pozwala na dodanie nowych słów do słownika istniejącego modelu, a jedynie na dalszą naukę w oparciu o słowa już istniejące w słowniku. Siłą rzeczy model zbudowany na zbiorze artykułów zawiera wszystkie występujące w artykułach słowa. Ostatecznie pozostałe modele metod *word embeddings*, których używam w dalszej części pracy są uczone i testowane na tym samym zbiorze artykułów *Allegro*.

Dokonuję porównania jakości modelu uczonego na korpusie *Słowosieci* z modelem uczonym na korpusie artykułów *Allegro*.

Sposób generowania wektorów i długość wektorów

Do wygenerowania wektorów reprezentujących słowa używam metod wektorowej reprezentacji tekstu: *Word2vec*, *GloVe* oraz *FastText*. Każdą z nich testuję pod kątem różnej wymiarowości generowanych wektorów.

Metoda określenia podobieństwa między wektorowymi reprezentacjami dokumentów

Zadaniem metod *word embeddings* jest wygenerowanie wektorowej reprezentacji słowa. W celu porównania całych dokumentów i wyznaczenia podobieństw między nimi należy użyć dodatkowych środków. W tym celu korzystam z poniższych metod:

1. odległość kosinusowa pomiędzy centroidami wektorowej reprezentacji tekstów (w skrócie metoda centroidu),
2. *Word Mover's Distance*,

3. autorska metoda wykorzystująca odległość kosinusową pomiędzy wektorową reprezentacją słów kluczowych wyznaczonych metodą *LDA*. Celem metody jest zmniejszenie wymiarowości artykułu oraz użycie tylko znaczących słów przy obliczaniu centroidu artykułu. W tej metodzie za pomocą *LDA* wyznaczam słowa kluczowe, które z największą wagą przynależą do danego artykułu. Nauczony model *LDA* każdemu dokumentowi i przypisuje zestaw tematów z określonymi wagami przynależności t_{ij} oraz każdemu tematowi j przypisuje zestaw słów wraz z wagami w_{jk}. Na tej podstawie dla każdego artykułu i mogę wybrać p słów, które posiadają najwyższą wagę przynależności do artykułu liczoną wg. wzoru $x(i,k) = \sum_j t_{ij} * w_{jk}$, gdzie k to dane słowo. Ostatecznie artykuł reprezentuję jako centroid n słów o najwyższym x dla danego słowa. W dalszych testach przyjmuję $n = 10$.

Powyższe metody stosuję w celu wyznaczenia dla danego artykułu a p artykułów najbardziej do niego podobnych uszeregowanych malejąco pod względem relewantności do artykułu bazowego a.

Elasticsearch

W celu porównania z metodami semantycznymi testuję również dotychczasowe zapytanie do silnika *Elasticsearch* używane dotychczas w *Allegro*. Zapytanie to odpowiada listą p artykułów najbardziej zbliżonych do artykułu wejściowego w kolejności od najbardziej podobnego.

Metoda losowa

W celu uzyskania punktu odniesienia dokonuję testów dla metody losowo wybierającej p podobnych do danego artykułów. Wybór następuje zgodnie z rozkładem jednostajnym ze zbioru wszystkich artykułów.

Metody ewaluacji

W celu wykonania oceny testowanych metod wykorzystuję wszystkie metody ewaluacji opisane w rozdziale 4, które opatruję skróconymi sygnaturami. Wszystkie miary przyjmują wartości $[0, 1]$.

1. *clicks* — ocena na podstawie historycznej aktywności użytkowników mierzona na podstawie liczby kliknięć w odnośniki.

2. *mut_ cat[_ ndcg]* — relewantność wyszukanych artykułów liczona na podstawie liczby wspólnych kategorii z artykułem bazowym. Stosuję dwa warianty: średnia relewantność wyszukanych artykułów oraz miara *nDCG*.

3. *mut_kw[_ndcg]* — relewantność wyszukanych artykułów liczona na podstawie liczby wspólnych słów kluczowych z artykułem bazowym. Również stosuję dwa warianty: średnia relewantność wyszukanych artykułów oraz miara *nDCG*.

4. *users* — ocena na podstawie eksperckiej oceny użytkowników. W badaniu wykorzystałem 5 użytkowników operujących każdy na tym samym zbiorze par testowych. Pary zostały wygenerowane (zgodnie z wcześniejszym opisem metody) na podstawie 50 artykułów bazowych wylosowanych spośród wszystkich artykułów udostępnionych mi przez *Allegro*.

Wyniki badań

W rozdziale tym przedstawiam wyniki oceny jakości poszczególnych metod w zależności od ich hiperparametrów. Wyniki te zebrane są w postaci tabel zawierających oryginalne wartości opisanych uprzednio miar ewaluacji. Każdy wiersz zawiera wyniki danej miary dla różnych konfiguracji testowanej metody. Każda kolumna zawiera wyniki ewaluacji danej konfiguracji metody uzyskane przez każdą z miar. Stąd dane należy analizować jedynie w obrębie wiersza, gdyż wyniki uzyskane różnymi miarami nie są porównywalne — powstały na różne sposoby. W każdym wierszu pogrubieniem oznaczam najwyższy wynik uzyskany przy użyciu danej miary. Kolumna, w której się on znajduje reprezentuje najlepszą konfigurację metody wg. tej miary.

W celu zwiększenia czytelności przedstawiam na wykresach znormalizowane wyniki z tabel. Normalizacja ma na celu sprawić, aby wyniki uzyskane różnymi miarami mogły być przedstawione wspólnie i były porównywalne ze sobą. Normalizacja na wykresach odbywa się przez podzielenie wszystkich wyników uzyskanych daną miarą przez najwyższy z nich.

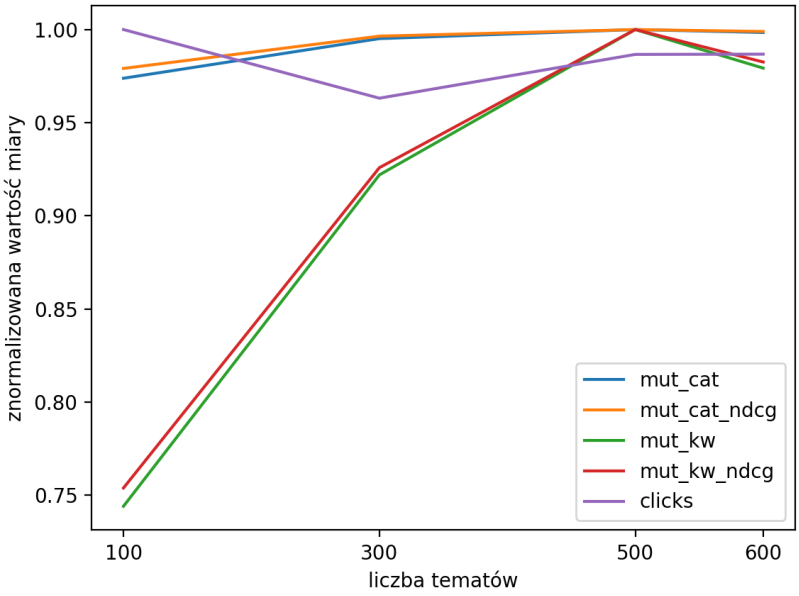
Do wyników każdego testu dołączam ich interpretację oraz wnioski.

LSI w zależności od liczby tematów

Dokonuję porównania jakości modelu *LSI* w zależności od wartości hiperparametru — liczby tematów — przy użyciu opisanych wcześniej miar.

liczba tematów	100	300	500	600
mut_cat	0.4289	0.4383	**0.4404**	0.4397
mut_cat_ndcg	0.5072	0.5162	**0.5181**	0.5175
mut_kw	0.2961	0.3669	**0.3980**	0.3898
mut_kw_ndcg	0.0973	0.1195	**0.1291**	0.1268
clicks	**0.9255**	0.8914	0.9132	0.9133

Tablica 5.1: Wyniki ewaluacji metody *LSI* różnymi miarami dla zmiennej liczby tematów.



Rysunek 5.1: Wykres porównujący znormalizowane wyniki różnych miar dla rosnącej liczby tematów metody *LSI*.

Miary oparte na liczbie wspólnych słów kluczowych (*mut_kw...*) wyraźnie pokazują wzrost jakości modelu przy wzroście liczby tematów. Wskazują one, że najlepsze rezultaty metoda *LSI* osiąga dla liczby tematów 500, po czym dla wyższej liczby tematów jakość przestaje się poprawiać.

Natomiast wyniki miar *clicks* i *mut_cat...* nie różnią się dla różnych liczb tematów więcej niż o 5%, stąd na ich podstawie nie da się wybrać najlepszej konfiguracji.

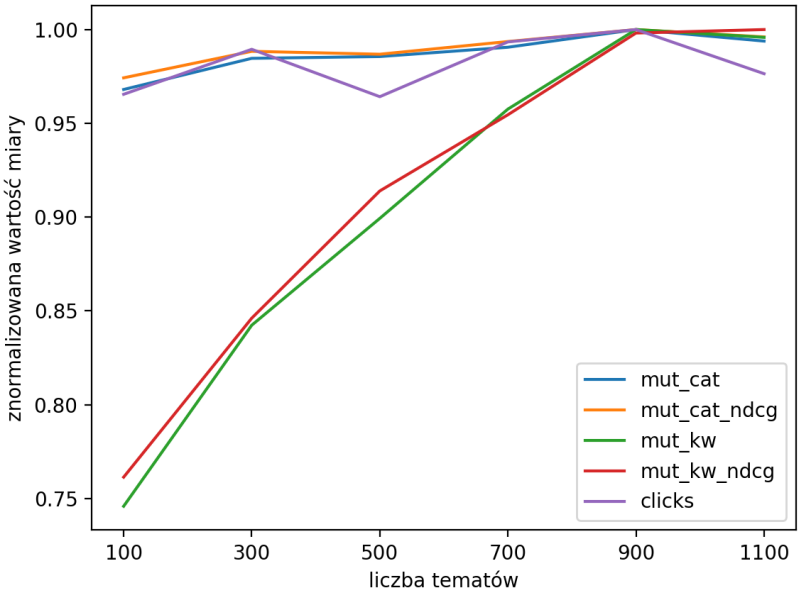
Warto zauważyć wyraźny związek pomiędzy wynikami dla wariantów metod opartych o średnią relewantność (np. *mut_kw*) oraz opartych o metodę *nDCG* (np. *mut_kw_ndcg*).

LDA w zależności od liczby tematów

Podobnie jak w poprzednim punkcie dokonuję porównania jakości modelu *LDA* w zależności od wartości hiperparametru — liczby tematów — przy użyciu wprowadzonych przez siebie miar.

liczba tematów	100	300	500	700	900	1100
mut_cat	0.4114	0.4184	0.4188	0.4210	**0.4250**	0.4223
mut_cat_ndcg	0.4879	0.4950	0.4942	0.4976	**0.5008**	0.4989
mut_kw	0.2196	0.2480	0.2648	0.2819	**0.2944**	0.2932
mut_kw_ndcg	0.0734	0.0815	0.0881	0.092	0.0962	**0.0964**
clicks	0.8809	0.9028	0.8797	0.9064	**0.9124**	0.8909

Tablica 5.2: Wyniki ewaluacji metody *LDA* różnymi miarami dla zmiennej liczby tematów.



Rysunek 5.2: Wykres porównujący znormalizowane wyniki różnych miar dla rosnącej liczby tematów metody *LDA*.

Podobnie jak w przypadku modelu *LSI* miary oparte na liczbie wspólnych słów kluczowych (*mut_kw...*) pokazują wzrost jakości modelu przy wzroście liczby tematów. Według tych miar najwyższą jakość model osiąga dla liczby tematów 900. W tym modelu miary oparte o kategorie (*mut_cat...*) wskazują najlepszy rezultat dla liczby tematów 900-1100, ale ich niskie wahania (<5%) sugerują, że ich wyniki mogą nie być dla tego przypadku miarodajne. Podobnie oceniam wyniki miary *clicks*, która nie wykazuje powiązania liczby tematów z jakością modelu.

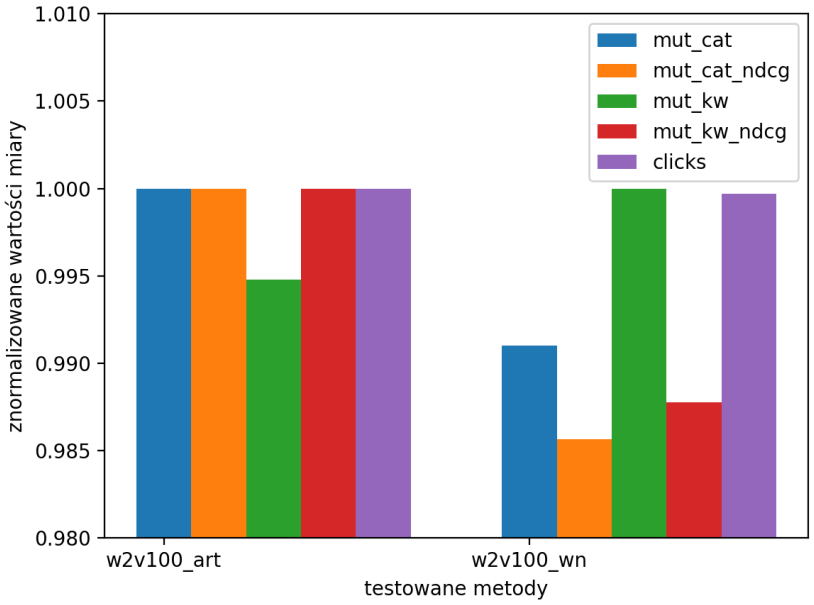
Warto zauważyć, że obie metody oparte o ekstrakcję tematów — *LSI* oraz *LDA* — osiągają najlepsze rezultaty dla różnych liczb tematów. Wynika to zapewne z faktu różnych sposobów wyznaczania owych tematów oraz różnej ich postaci w każdej z metod.

Word2vec w zależności od korpusu

Poniżej zestawiam wyniki dla modelów *Word2vec* o wektorach długości 100 uczonych: na korpusie artykułów z *Allegro* (*w2v100_art*) oraz na korpusie *Słowosieci* (*w2v100_wn*). Dokumenty porównywane są tu metodą centroidu.

sygnatura metody	w2v100_art	w2v100_wn
mut_cat	**0.4267**	0.4228
mut_cat_ndcg	**0.5040**	0.4968
mut_kw	0.2770	**0.2784**
mut_kw_ndcg	**0.0896**	0.0885
clicks	**0.9465**	0.9462

Tablica 5.3: Wyniki ewaluacji metody *Word2vec* w zależności od korpusu, na którym trenowany był model.



Rysunek 5.3: Wykres porównujący znormalizowane wyniki dla metody *Word2vec* w zależności od bazowego korpusu.

Mimo, iż wykres pozornie przedstawia duże różnice między metodami, to maksymalna różnica między najlepszym i najgorszym wynikiem dla dowolnej z miar nie przekracza 2%. Stąd trudno jest stwierdzić, metoda uczona na którym korpusie daje tu lepsze wyniki. Na korzyść modelu *Word2vec* uczonego na korpusie artykułów przemawia jego mały rozmiar, a co za tym idzie szybkość nauki. Z przyczyny braku wyraźnych zalet korzystania tu z modelu uczonego na korpusie *Słowosieci* korzystam dalej jedynie z modeli opartych o korpus artykułów *Allegro*.

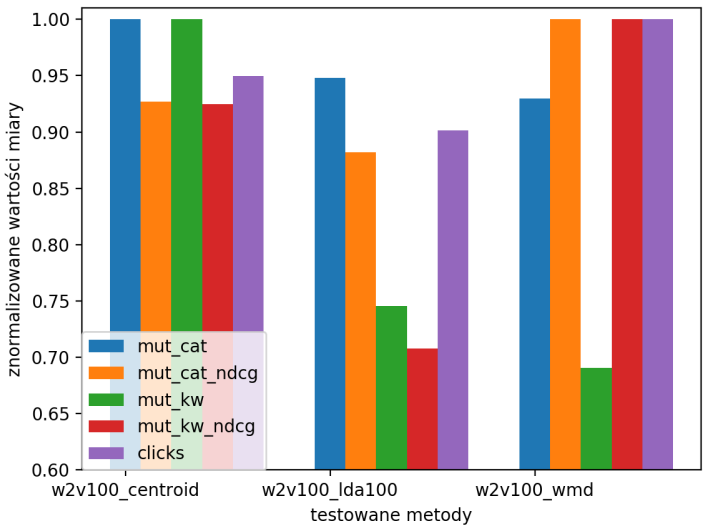
Word2vec w zależności od metody porównywania dokumentów

W niniejszym punkcie dokonuję porównania jakości metod określania podobieństwa między wektorowymi reprezentacjami dokumentów tekstowych wyznaczonymi metodą *Word2vec* dla wektorów długości 100 oraz 300. Biorę tu pod uwagę metody: centroidu całości dokumentu (*w2v100_centroid*), centroidu słów kluczowych wyznaczonych metodą *LDA* dla 100 tematów (*w2v100_lda100*) oraz *Word Mover's Distance* (*w2v100_wmd*). Z powodu powolności metody *WMD* stosuję ją do wyznaczenia dokumentów najbliższych danemu jedynie spośród 20 dokumentów najbardziej podobnych do danego wyznaczonych wcześniej metodą centroidu.

Wymiar wektorów: 100

sygnatura metody	w2v100_centroid	w2v100_lda100	w2v100_wmd
mut_cat	**0.4267**	0.4044	0.3968
mut_cat_ndcg	0.5040	0.4797	**0.5439**
mut_kw	**0.2770**	0.2065	0.1913
mut_kw_ndcg	0.0896	0.0686	**0.0969**
clicks	0.9465	0.8985	**0.9967**

Tablica 5.4: Wyniki ewaluacji metody *Word2vec* różnymi miarami dla wektorów dł. 100 w zależności od metody porównywania wektorowych reprezentacji dokumentów.

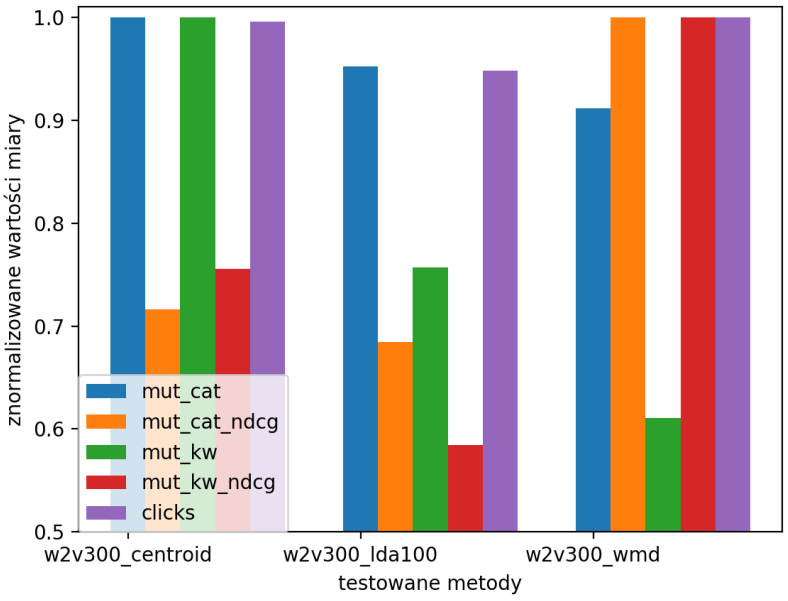


Rysunek 5.4: Porównanie znormalizowanych wyników dla metody *Word2vec* w zależności od sposobu wyznaczania podobieństwa między wektorowymi reprezentacjami dokumentów.

Wymiar wektorów: 300

sygnatura metody	w2v300_ctr	w2v300_lda100	w2v300_wmd
mut_cat	**0.4277**	0.4072	0.3898
mut_cat_ndcg	0.5055	0.4829	**0.7055**
mut_kw	**0.2807**	0.2124	0.1714
mut_kw_ndcg	0.0907	0.0701	**0.1200**
clicks	0.9364	0.8916	**0.9406**

Tablica 5.5: Wyniki ewaluacji metody *Word2vec* różnymi miarami dla wektorów dł. 300 w zależności od metody porównywania wektorowych reprezentacji dokumentów.



Rysunek 5.5: Porównanie znormalizowanych wyników dla metody *Word2vec* w zależności od sposobu wyznaczania podobieństwa między wektorowymi reprezentacjami dokumentów.

Z powyższych testów wprost wynika, że rezultaty dla metody opartej o tematy *LDA* są jednoznacznie gorsze od wyników dla metody centroidu.

Dla obu konfiguracji metody *Word2vec* metoda *Word Mover's Distance* wykazuje się największą zdolnością do poprawnego szeregowania dokumentów (wysoki wskaźnik metod *nDCG* w porównaniu z metodami wykorzystującymi średnią).

Wadą metody *WMD* wykluczającą ją z użycia w tym przypadku jest jej powolność. Dla pierwszego przypadku obliczenia trwały 55 minut, co przy czasie <1 sek dla centroidu jest wartością

niedopuszczalną. Proporcjonalnie użycie tej metody dla całego korpusu (a nie jedynie dla 20 artykułów wybranych metodą centroidu) trwałoby ok. 183 godzin.

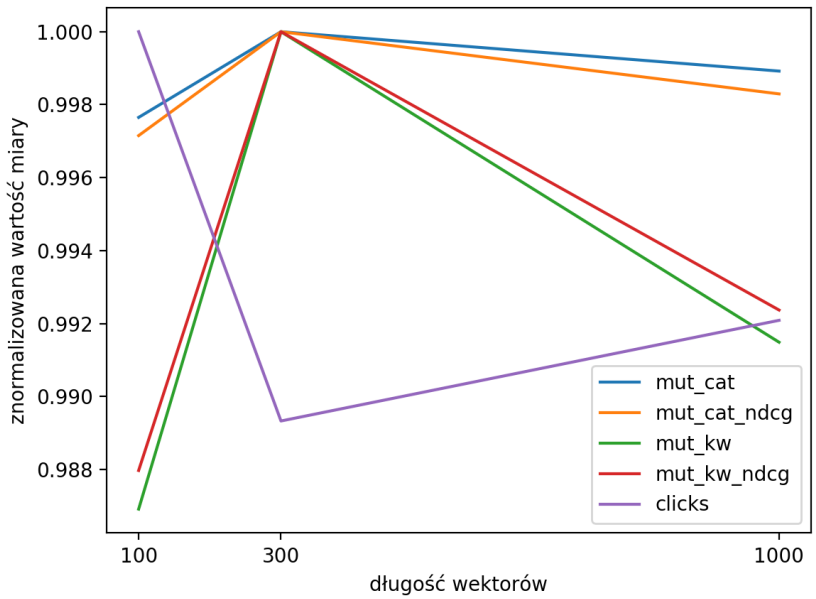
Metody *word embeddings* w zależności od wymiarowości wektorów

W niniejszym punkcie porównuję jakość metod *word embeddings* (*Word2vec*, *GloVe* i *FastText*) w zależności od wymiarowości wektorów wyjściowych: 100, 300 i 1000. Użyta metoda wyznaczania podobieństwa dokumentów to metoda centroidu.

Word2vec

długość wektorów	100	300	1000
mut_cat	0.4267	**0.4277**	0.4272
mut_cat_ndcg	0.5040	**0.5055**	0.5046
mut_kw	0.2770	**0.2807**	0.2783
mut_kw_ndcg	0.0896	**0.0907**	0.0900
clicks	**0.9465**	0.9364	0.939

Tablica 5.6: Wyniki ewaluacji metody *Word2vec* dla zmiennej długości wektorów.



Rysunek 5.6: Porównanie znormalizowanych wyników dla metody *Word2vec* w zależności od hiperparametru — długości wektorów.

Dla różnej wymiarowości wektorów wyjściowych metoda *Word2vec* daje bardzo podobne rezultaty. Różnica między najlepszym i najgorszym wynikiem wynosi jedynie ok. 1,5%. Mimo to najlepszą wartością hiperparametru wydaje się być 300 — wskazują na to miary *mut_cat...* oraz *mut_kw...*.

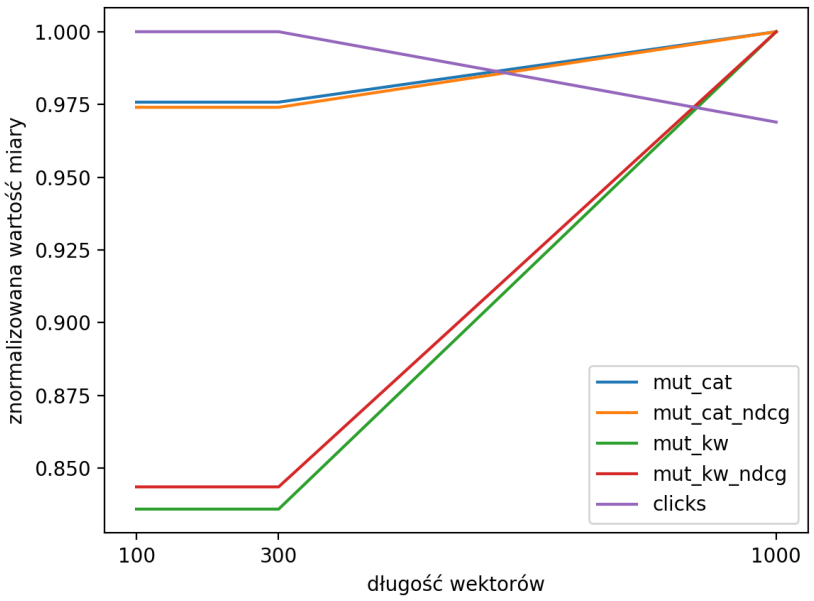
Wyraźnie widoczne są natomiast odstające wyniki metody *clicks*. Można

to tłumaczyć niedoskonałością tej miary opartej o preferencje realnych użytkowników serwisu. Szerszej interpretacji zjawiska dokonuję w podsumowaniu rozdziału.

GloVe

długość wektorów	100	300	1000
mut_cat	0.4324	0.4324	**0.4432**
mut_cat_ndcg	0.5101	0.5101	**0.5237**
mut_kw	0.2998	0.2998	**0.3586**
mut_kw_ndcg	0.0971	0.0971	**0.1151**
clicks	**0.908**	**0.908**	0.8798

Tablica 5.7: Wyniki ewaluacji metody *GloVe* dla zmiennej długości wektorów.



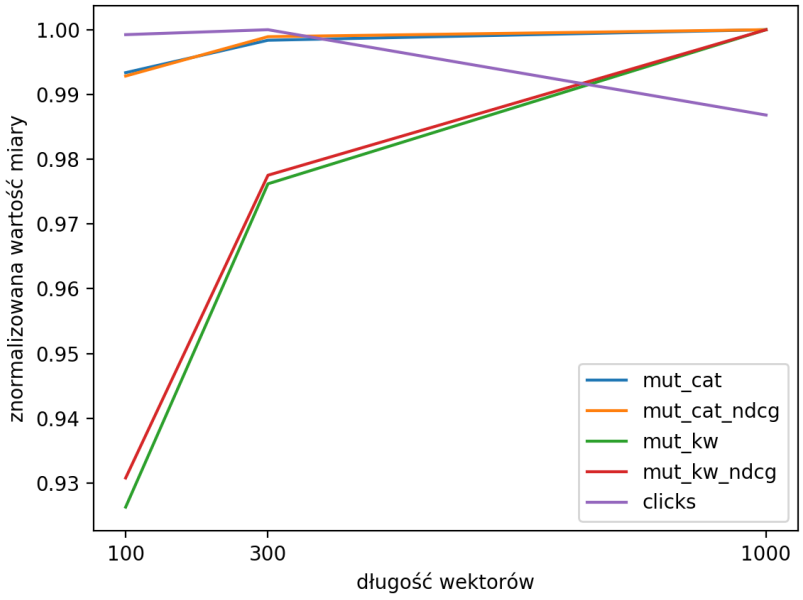
Rysunek 5.7: Porównanie znormalizowanych wyników dla metody *GloVe* w zależności od hiperparametru — długości wektorów.

Ewidentnie najlepsze wyniki metoda osiąga dla wektorów wyjściowych długości 1000. Świadczą o tym zgodne wysokie wyniki wszystkich miar prócz miary *clicks*.

FastText

długość wektorów	100	300	1000
mut_cat	0.4444	0.4466	**0.4473**
mut_cat_ndcg	0.5252	0.5284	**0.5289**
mut_kw	0.335	0.353	**0.3617**
mut_kw_ndcg	0.1085	0.1139	**0.1165**
clicks	0.8968	**0.8974**	0.8856

Tablica 5.8: Wyniki ewaluacji metody *FastText* dla zmiennej długości wektorów.



Rysunek 5.8: Porównanie znormalizowanych wyników dla metody *FastText* w zależności od hiperparametru — długości wektorów.

Podobnie jak w przypadku metody *GloVe* metoda *FastText* osiąga najlepsze rezultaty dla wektorów długości 1000. Również wyniki poszczególnych miar wyglądają bardzo podobnie do wyników przy poprzednio testowanej metodzie *GloVe*

Ocena jakości wybranych metod przez użytkowników

W niniejszym punkcie dokonuję porównania najlepszych konfiguracji testowanych wcześniej metod. Zestawiam z nimi wyniki dla metody używanej dotychczas w *Allegro* oraz metodę losową. Testowane warianty:

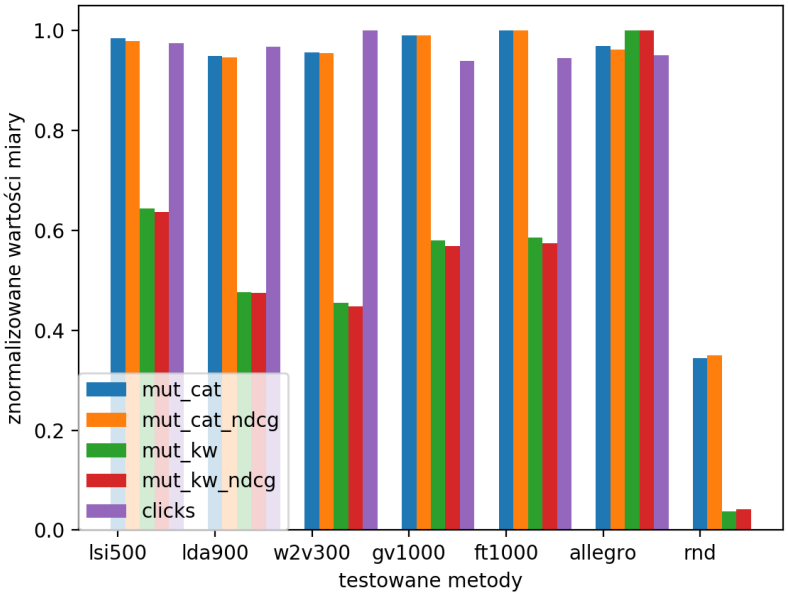
- *lsi*500 — *Latent Semantic Indexing* dla 500 tematów,
- *lda*900 — *Latent Dirichlet Allocation* dla 900 tematów,
- *w*2*v*300 — *Word2vec* dla wektorów dł. 300 i przy użytej metodzie centroidu,
- *w*2*v*300 — *GloVe* dla wektorów dł. 1000 i przy użytej metodzie centroidu,
- *w*2*v*300 — *FastText* dla wektorów dł. 1000 i przy użytej metodzie centroidu,
- *allegro* — używane dotychczas w*Allegro* zapytanie do silnika *Elasticsearch*,
- *rnd* — metoda losowa.

Zestawienie wyników testów automatycznych

Poniżej zestawienie wyników miar automatycznych dla najlepszych wariantów testowanych wcześniej metod.

sygnatura metody	lsi500	lda900	w2v300	gv1000	ft1000	allegro	rnd
mut_cat	0.4404	0.4250	0.4277	0.4432	**0.4473**	0.4339	0.1541
mut_cat_ndcg	0.5181	0.5008	0.5055	0.5237	**0.5289**	0.5091	0.1854
mut_kw	0.3980	0.2944	0.2807	0.3586	0.3617	**0.6175**	0.0229
mut_kw_ndcg	0.1291	0.0962	0.0907	0.1151	0.1165	**0.2027**	0.0084
clicks	0.9132	0.9124	**0.9364**	0.8798	0.8856	0.8904	—

Tablica 5.9: Zestawienie wyników testów automatycznych dla najlepszych konfiguracji wybranych metod.



Rysunek 5.9: Porównanie znormalizowanych wyników dla wybranych metod.

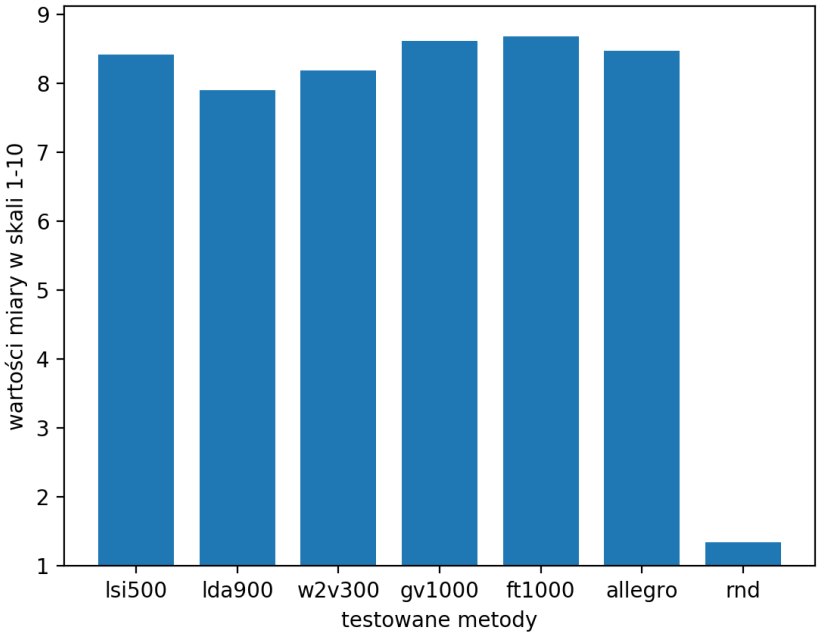
Wszystkie testowane metody dają dużo lepsze rezultaty od metody losowej. Stosunkowo wysokie wskaźniki *mut_cat* dla metody losowej wynikają z dużo większej szansy, że dwa artykuły mają wspólnych przodków w drzewie kategorii niż, że mają wspólne słowa kluczowe.

Najwyższe wyniki wszystkich miar uzyskała metoda *allegro*. Jej wysokie wskaźniki dla metody *mut_kw* wynikają wprost ze struktury zapytania do silnika *Elasticsearch*, które to jest oparte właśnie o słowa kluczowe dołączone do artykułu. Poza tym metoda ta ma bezpośredni wpływ na wyniki miary *clicks*, ponieważ generuje ona dotychczasowe rekomendacje w serwisie, a miara ta została opracowana na podstawie aktywności użytkowników właśnie w działającym dotychczas serwisie.

Wyniki oceny eksperckiej

sygnatura metody	lsi500	lda900	w2v300	gv1000	ft1000	allegro	rnd
users	8.4150	7.895	8.1850	8.6100	**8.6800**	8.4700	1.345

Tablica 5.10: Zestawienie wyników ewaluacji eksperckiej dla najlepszych konfiguracji wybranych metod.



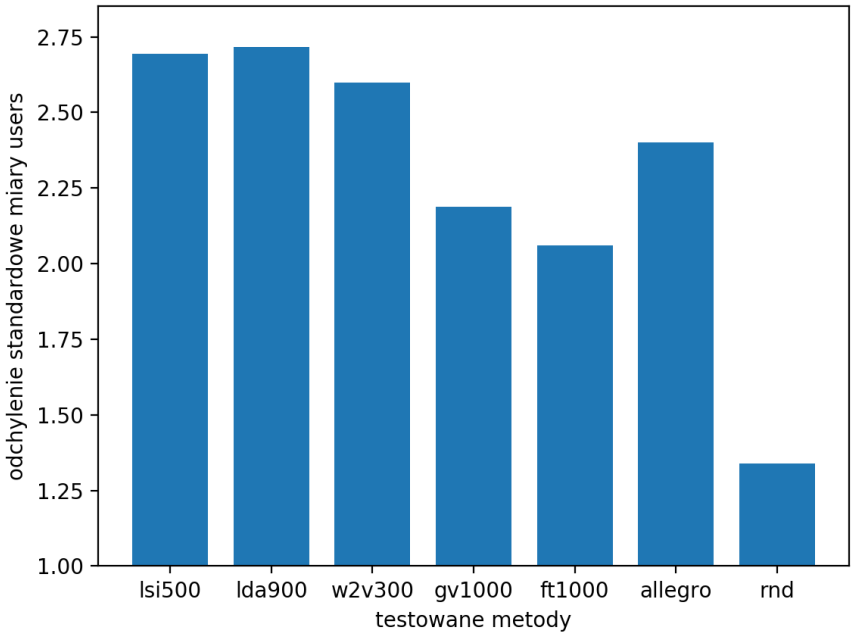
Rysunek 5.10: Porównanie wyników ewaluacji eksperckiej dla wybranych metod.

Ewaluacja ekspercka wprost pokazuje, że żadna z testowanych metod nie odbiega znacząco od innych. Żadna z testowanych adaptacji metod semantycznej analizy tekstu nie odbiega również od dotychczasowej metody używanej w *Allegro*. Różnica pomiędzy najlepszą metodą (*FastText*) i najgorszą (*LDA*) wynosi ok. 9%. Każda z testowanych metod dała również wynik znacząco lepszy od metody losowej.

Pożądaną cechą dla metody jest, aby możliwie rzadko generowała ona wyniki odstające — w żaden sposób niezwiązane z artykułem bazowym. Stąd niska wartość odchylenia standardowego działa na korzyść metody. Tutaj również najlepszy rezultat — najbardziej spójne wyniki uzyskała metoda *FastText*, natomiast najgorszy wynik znów uzyskała metoda *LDA*.

sygnatura metody	lsi500	lda900	w2v300	gv1000	ft1000	allegro	rnd
users_std	2.6931	2.7155	2.5982	2.1881	2.0610	2.4019	1.3401

Tablica 5.11: Zestawienie odchyleń standardowych ocen dokonanych przy ewaluacji eksperckiej dla najlepszych konfiguracji wybranych metod.



Rysunek 5.11: Porównanie odchyleń standardowych ocen eksperckich dla wybranych metod.

W świetle powyższych wyników najlepsze rezultaty osiąga *FastText*. Zakładam, iż jest to związane z wewnętrzną analizą słów, jaką wykonuje ta metoda. Może się to sprawdzać szczególnie dobrze w przypadku języka polskiego, który jest językiem bogatym morfosyntaktycznie.

Aby odpowiedzieć na pytanie postawione na początku niniejszej pracy, tj. czy metody semantycznej analizy języka naturalnego zaadaptowane w celu wyznaczania rekomendacji są w stanie dorównać jakością dotychczasowej metodzie stosowanej w *Allegro*, wykonuję test, który ma na celu sprawdzenie, czy są podstawy, by sądzić, że wyniki którejkolwiek z metod poddanych ocenie eksperckiej odstają w sposób istotny statystycznie od reszty przetestowanych metod.

Test statystyczny, jaki przeprowadzam, to test Kruskala-Wallisa. Za hipotezę zerową H_0 przyjmuje on równość dystrybuant rozkładów w populacjach, z których pochodzą próby.

We wcześniejszych testach użytkownicy oceniali podobieństwo pomiędzy artykułem, a każdym z artykułów do niego rekomendowanych, wyznaczonych ewaluowanymi metodami. Za dane wejściowe testu Kruskala-Wallisa przyjmuję próby odpowiadające wynikom testów dla każdej z metod (prócz metody losowej) złożone ze średniej oceny użytkowników dla najbardziej relewantnej rekomendacji do każdego z testowanych artykułów bazowych.

W teście przyjmuję poziom istotności $\alpha = 0.05$. Ostatecznie, jako wynik testu Kruskala-Wallisa otrzymuję $p = 0.0571 > \alpha$, na podstawie czego stwierdzam, że brak jest przesłanek, by odrzucić hipotezę zerową — różnica między jakością rekomendacji testowanych metod nie jest

istotna statystycznie. Oznacza to, że ani żadna z testowanych metod semantycznych nie odstaje od innych, ani też dotychczasowa metoda *Allegro* nie jest istotnie lepsza/gorsza od innych testowanych metod.

Podsumowanie testów

Przeprowadzone testy automatyczne miały na celu sprawdzenie, jak jakość danej metody zależy od jej konfiguracji. W tym celu wykorzystałem autorskie metody ewaluacji. W wielu przypadkach dało się zauważyć korelację między metodami ewaluacji opartymi o metadane artykułów: słowa kluczowe i kategorie.

Wartościowym okazało się rozróżnienie na ewaluację poprzez obliczenie średniej relewantności oraz obliczenie miary *nDCG*. Pokazało to wyjątkowo dobrą zdolność metody *WMD* do poprawnego szeregowania rekomendowanych artykułów.

Wyniki metody *clicks* — opartej o historyczną aktywność użytkowników — wyraźnie odstają od innych wyników. Może się to wiązać z faktem, iż liczba przejść dokonanych przez użytkowników między parami artykułów niekoniecznie odzwierciedla podobieństwo między nimi. Fakt, iż użytkownik przechodzi z artykułu A do rekomendowanego artykułu B może również wynikać z faktu, iż artykuł B porusza inną tematykę niż artykuł A, która to jednak interesuje użytkownika. Wprowadzanie do rekomendacji małego odsetka niepowiązanych przedmiotów jest znaną praktyką przy tworzeniu systemów rekomendacji i ma na celu zainteresowanie użytkownika przedmiotami spoza kręgu jego dotychczasowych zainteresowań.

Porównanie najlepszych konfiguracji testowanych metod wykazało, że nie ma istotnych różnic między metodami semantycznej analizy tekstu zaadaptowanymi do zadania generowania rekomendacji oraz dotychczasową metodą stosowaną w *Allegro* opartą o zapytanie do silnika *Elasticsearch*.

Podsumowanie pracy i kierunki dalszych badań

W niniejszej pracy starałem się sprawdzić, czy metody semantycznej analizy tekstu zaadaptowane w celu wyznaczania artykułów najbardziej podobnych do danego są w stanie dorównać dotychczas używanej w serwisie *Allegro* metodzie opartej o zapytanie do silnika wyszukiwania *Elasticsearch*. W tym celu wykonałem szereg testów sprawdzających jakość wygenerowanych rekomendacji w zależności od konfiguracji poszczególnych metod. Po analizie wyników testów można udzielić odpowiedzi: tak, każda z metod semantycznej analizy języka naturalnego w swojej najlepszej konfiguracji może zostać zaadaptowana w celu generowania rekomendacji artykułów w oparciu o treść artykułu aktualnie wyświetlanego przez użytkownika serwisu. Rekomendacje te w ocenie użytkowników nie ustępują jakością dotychczasowej metodzie.

Użycie metod semantycznej analizy tekstu pozwala wychwycić ukryte podobieństwa między dokumentami, gdzie dokumenty łączą nie te same słowa, czy synonimy, ale pewne złożone abstrakcyjne koncepty powiązane ze sobą. Istotną zaletą metod semantycznych w porównaniu z dotychczasową metodą wykorzystywaną w *Allegro* jest fakt, iż opierają się one wyłącznie na treści artykułów. Zwalnia to autorów artykułów z samodzielnego opatrywania ich dodatkowymi metatagami, z których w dużym stopniu korzysta dotychczasowa metoda.

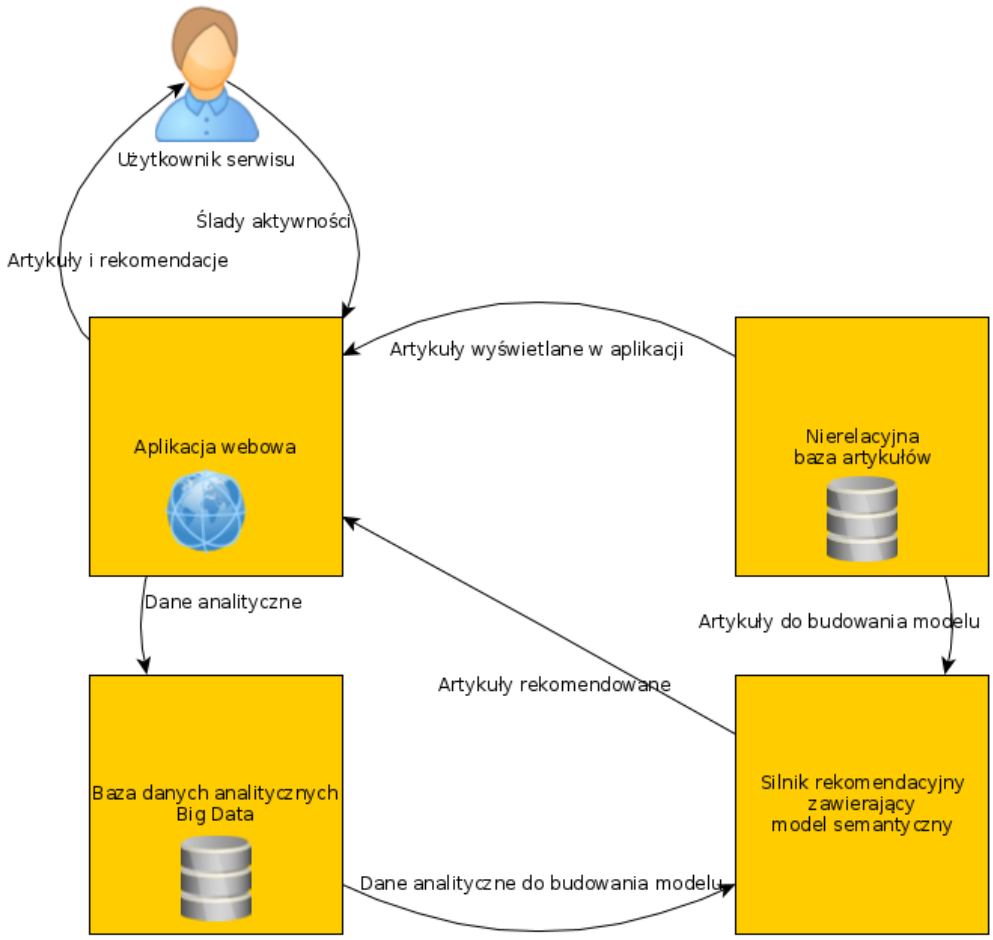
Z kolei istotną zaletą dotychczasowego rozwiązania stosowanego w *Allegro* jest uniwersalność silnika *Elasticsearch* oraz to, że pozwala edytować indeksowane dane w locie, bez konieczności przebudowy systemu. Metody semantycznej analizy tekstu wymagają przebudowania modelu w celu jego aktualizacji przy zmieniających się danych.

Analiza zawarta w tej pracy nie wyczerpuje całkowicie zagadnień związanych z tematem pracy. Przeprowadzone w pracy badania dają jedynie ogólny pogląd, ale pokazują, że istnieje pole do dalszych badań w tym kierunku.

Nietrywialnym problemem okazało się zadanie ewaluacji systemu rekomendacyjnego. Warto zaznaczyć tu różnicę pomiędzy tematyką pracy, a komercyjnym zagadnieniem najlepszych rekomendacji. Artykuły najbardziej podobne wcale nie muszą być tymi, które najlepiej nadają się na rekomendacje. Otwartym pytaniem pozostaje, czy rekomendowanie użytkownikowi podobnych

artykułów wprost przekłada się na zwiększenie liczby transakcji zawartych przez niego za pośrednictwem serwisu. Powszechnym zjawiskiem jest wzbogacanie rekomendacji o przedmioty niepodobne do danego, a pozwalające użytkownikowi na poznanie osobnej kategorii przedmiotów, która może go zainteresować, a tym samym przyciągnąć do serwisu. Ostatecznym celem użycia systemu rekomendacji jest w dłuższej perspektywie maksymalizacja zysku serwisu przez kierowanie użytkowników na strony ofert, co zwiększa szansę na dokonanie przez nich transakcji. Trudność oceny jakości takiego systemu wiąże się z mnogością czynników, które kierują zachowaniami użytkowników. Dopiero badania na „żywym systemie" są w stanie wykazać realną wyższość jednej z metod nad innymi. Sporządzenie systemu rekomendacji uwzględniającego te czynniki wydaje się więc być osobnym tematem godnym szczegółowych badań.

Badania przeprowadzone w niniejszej pracy mogą być podstawą do zaprojektowania pełnego systemu rekomendacji, który mógłby zostać wykorzystany w realnej aplikacji.



Rysunek 6.1: Ogólny schemat aplikacji wykorzystującej system rekomendacyjny oparty na treści artykułów.

W aplikacji tej użytkownik odpytuje aplikację za pośrednictwem strony internetowej w celu wyświetlenia treści artykułu. Gdy porusza się po stronach serwisu, informacje o sekwencji odwiedzonych przez niego stron zostają zapisane w bazie danych analitycznych. Artykuły wyświetlane przez użytkownika pochodzą z bazy danych o zoptymalizowanym odczycie — np. silnika wyszukiwania. Z założenia edycja takiej bazy następuje nieporównywalnie rzadziej niż odczyt z niej. Do każdego wyświetlanego przez użytkownika artykułu dołączane są artykuły do niego rekomendowane. Wybór artykułów rekomendowanych dokonywany jest przez silnik rekomendacyjny oparty o model semantycznej analizy języka naturalnego. Model ten jest uczony na podstawie artykułów pochodzących z bazy artykułów. Jego ewaluacja następuje na podstawie danych analitycznych gromadzonych w osobnej bazie. W trakcie działania systemu zmieniają się zachowania użytkowników, warstwa wizualna stron internetowych oraz zbiór dostępnych artykułów. W związku z tym model musi być co pewien czas przebudowywany w celu maksymalizacji jakości generowanych wyników.

Przed zastosowaniem metod wyznaczania podobieństwa wykonałem przetwarzanie wstępne dokumentów, które można przeprowadzić również na inne sposoby. Dobór metod wstępnego przetwarzania tekstu jest zawsze kwestią indywidualną, trudną w ocenie.

Można by również zastosować inne metody wyznaczania odległości między wektorowymi reprezentacjami dokumentów. Metoda centroidu jest szybka i prosta, ale dokonuje sporego uproszczenia struktury dokumentu. Metoda *WMD* daje natomiast lepsze uszeregowanie, ale kosztem dużej złożoności obliczeniowej.

Bibliografia

[1] Opis Allegro `https://magazyn.allegro.pl/3333-serwis-allegro-to-nasz-sposob-na-wasze-szybkie-i-wygodne-zakupy-przez-internet` (dostęp 07.05.2017)

[2] Y. Bengio, R. Ducharme, P. Vincent, C. Jauvin, *A Neural Probabilistic Language Model*, Journal of Machine Learning Research 3 1137–1155, 2003

[3] D. M. Blei, A. Y. Ng, M. I. Jordan, *Latent Dirichlet Allocation*, Journal of Machine Learning Research, tom 3 num. 4–5, 2003

[4] Blog Aylien `http://blog.aylien.com/overview-word-embeddings-history-word2vec-cbow-glove/` (dostęp 18.08.2017)

[5] R. Collobert, J. Weston, *A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning*, NEC Labs America, 2008

[6] W. B. Croft, D. Metzler, T. Strohman, *Search Engines. Information Retrieval in Practice*, Pearson Education, Inc., 6-9, 2015

[7] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, R. Harshman, *Indexing by latent semantic analysis*, Journal of the American Society for Information Science, tom 41, num. 6, 1990

[8] Elasticsearch `https://www.elastic.co/` (dostęp 18.08.2017)

[9] Firmy korzystające z Elasticsearch `https://www.elastic.co/use-cases` (dostęp 10.08.17)

[10] J.R. Firth, *A synopsis of linguistic theory 1930-1955*, Oxford: Philological Society, 1957

[11] G. H. Golub, W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis. 2 (2), 1965

[12] Z. S. Harris, *Distributional Structure*, Word, 10 (2/3): 146–62, 1954

[13] Informacje o Word2vec `https://code.google.com/archive/p/word2vec/` (dostęp 26.05.2017)

[14] K. Jarvelin, J. Kekalainen, *Cumulated gain-based evaluation of IR techniques*, University of Tampere, 2002

[15] A. Joulin, E. Grave, P. Bojanowski T. Mikolov, *Bag of Tricks for Efficient Text Classification*, Facebook AI Research, 2016

[16] P. Kędzia, G. Czachor, M. Piasecki, J. Kocoń *Vector representations of polish words (Word2Vec method)*, Wrocław University of Technology, 2016, `https://clarin-pl.eu/dspace/handle/11321/327` (dostęp 26.06.2017)

[17] M. J. Kusner, Y. Sun, N. I. Kolkin, K. Q. Weinberger, *From Word Embeddings To Document Distances*, International Conference on Machine Learning (ICML), 2015

[18] Lucene `https://lucene.apache.org/` (dostęp 18.08.2017)

[19] T. Mikolov, K. Chen, G. Corrado, J. Dean, *Efficient Estimation of Word Representations in Vector Space*, International Conference on Machine Learning (ICML), 2013

[20] Morfologik `http://morfologik.blogspot.com/` (dostęp 07.05.2017)

[21] Opis GloVe `https://cran.r-project.org/web/packages/text2vec/vignettes/glove.html` (dostęp 30.08.2017)

[22] Opis Word2vec `http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/` (dostęp 26.05.2017)

[23] O. Pele, M. Werman, *Fast and robust earth mover's distances*, ICCV, 2009

[24] J. Pennington, R. Socher, C. D. Manning, *GloVe: Global Vectors for Word Representation*, Computer Science Department, Stanford University, Stanford, CA 94305, 2014

[25] Porównanie największych polskich serwisów aukcyjnych `http://gadzetomania.pl/11824,zakupy-w-sieci-porownanie-najwiekszych-polskich-serwisow-aukcyjnych-2` (dostęp 09.08.17)

[26] F. Ricci, L. Rokach, B. Shapira, *Introduction to Recommender Systems Handbook*, Springer, 1-4, 2011

[27] F. Ricci, L. Rokach, B. Shapira, *Introduction to Recommender Systems Handbook*, Springer, 145-147, 2011

[28] F. Ricci, L. Rokach, B. Shapira, *Introduction to Recommender Systems Handbook*, Springer, 73-75, 2011

[29] Y. Rubner, C. Tomasi, L. J. Guibas, *The Earth Mover's Distance as a Metric for Image Retrieval*, Computer Science Department, Stanford University, 1, 2000

[30] G. Salton and M. McGill, *Introduction to modern information retrieval*, McGraw-Hill, 1983

[31] Słowa stopu w Wikipedii `https://pl.wikipedia.org/wiki/Wikipedia:Stopwords` (dostęp 15.04.2017)

[32] Słownik Języka Polskiego PWN `http://sjp.pwn.pl/sjp/artykul;2441396.html` (dostęp 07.05.2017)

[33] Strona Allegro z artykułem `https://allegro.pl/artykul/jaka-farba-dla-alergika-55917/` (dostęp 26.06.2017)

[34] Wordnet `http://plwordnet.pwr.wroc.pl/wordnet/` (dostęp 28.06.2017)

Spis rysunków

Spis tabel

Spis załączników

1. Wykorzystane technologie i narzędzia
2. Zawartość dołączonej płyty CD

Wykorzystane technologie i narzędzia

Analizę danych, ich wstępne przetworzenie, a następnie przeprowadzenie docelowych eksperymentów wykonałem korzystając głównie z języka *Python* i szeregu skryptów napisanych w nim własnoręcznie, wykorzystujących istniejące specjalistyczne biblioteki, posiadające interfejs w tymże języku. Poniżej przedstawiam wykorzystane narzędzia.

- *Elasticsearch* — silnik wyszukiwania tekstowego. Używam go do przechowywania bazy artykułów oraz ich przetworzonych wersji.
- *MongoDB* — nierelacyjna baza danych, której używam do przechowywania wyników generowanych przez testowane algorytmy.
- *GloVe* — implementacja metody *GloVe*. Generuje wektory słów, które następnie mogą być wykorzystane np. przez bibliotekę *gensim*.
- *FastText* — implementacja metody *FastText* wykonana przez zespół *Facebook Research*. Generuje wektory słów, które następnie mogą być wykorzystane np. przez bibliotekę *gensim*.

Poniższa lista zawiera wykorzystane biblioteki języka *Python*.

- *Gensim* — rozbudowana biblioteka służąca do przetwarzania języka naturalnego; zawiera implementację metod *Word2Vec*, *LSI*, *LDA*, *TF-IDF* i inne.
- *Morfologik* — tokenizer języka polskiego.
- *Numpy* — pozwala wydajnie wykonywać obliczenia numeryczne.
- *Scipy* — biblioteka do zastosowań naukowych, matematycznych i inżynierskich.
- *Pyemd* — implementacja algorytmu *Earth Mover's Distance*.
- *Elasticsearch* — ułatwia wykonywanie zapytań do silnika *Elasticsearch* wprost z kodu języka *Python*.
- *Matplotlib* — biblioteka służąca do wykonywania wykresów.

- *Pymongo* — umożliwia wykonywanie zapytań do bazy *MongoDB* wprost z kodu języka *Python*.

Zawartość dołączonej płyty CD

Do niniejszej pracy dołączam płytę CD zawierającą skrypty, których używałem do wykonania badań zawartych w niniejszej pracy. Skrypty dzielą się na 3 kategorie — są pogrupowane w foldery o następujących nazwach:

1. `similarity_methods` — adaptacje metod semantycznej analizy języka naturalnego do zadania generowania rekomendacji opartych na treści artykułu,
2. `preprocessing` — skrypty wykorzystywane przy wstępnym przetwarzaniu treści artykułów,
3. `evaluation_methods` — implementacje opracowanych metod ewaluacji.

Postać wykorzystywanych skryptów zmieniała się dynamicznie podczas prowadzenia badań, stąd ich obecna zawartość nie pokrywa wszystkich wykonanych przeze mnie zadań. Umieszczam je m.in. jako przykład wykorzystania użytych bibliotek. Nie da się ich uruchomić bez skonfigurowanego środowiska oraz, co ważniejsze, bez danych źródłowych.