2021 PRACE JĘZYKOZNAWCZE XXIII/3
ISSN 1509-5304 DOI 10.31648/pj.6839 105–113

Katarzyna Witkowska
Uniwersytet Warmińsko-Mazurski w Olsztynie
Ermlab Sp. z o.o.
ORCID: https://orcid.org/0000-0002-9603-607X
e-mail: katarzyna.witkowska@uwm.edu.pl

O projekcie kontekstowego rozumienia języka pisanego na potrzeby systemu automatycznej poprawy błędów dla języka polskiego*

The project of contextual understanding of the written language for the purpose of an error correction system designed for the Polish language

Abstrakt
Celem niniejszej publikacji jest przedstawienie projektu badawczo-rozwojowego związanego z opracowaniem technologii kontekstowego rozumienia języka pisanego na potrzeby systemu automatycznej poprawy błędów dla języka polskiego. Rozważania te w oczywisty sposób wpisują się w stosunkowo często podejmowany w ostatnim czasie (głównie na gruncie językoznawstwa angielskiego) nurt badań dotyczący wykorzystywania metod przetwarzania języka naturalnego (*natural language processing*) oraz uczenia maszynowego (*machine learning*) na potrzeby zaprojektowania systemów GEC (*grammatical error correction*). W pierwszym punkcie zaprezentowano (w ujęciu problemowym) stan badań, tj. najważniejsze koncepcje z zakresu sposobów tworzenia systemów GEC. Następnie przedstawiono charakter i cel prowadzonych prac badawczych oraz omówiono główne założenia metodologiczne. W przedostatnim paragrafie zasygnalizowano najważniejsze problemy związane z opracowaniem korpusu badawczego.

Słowa kluczowe: przetwarzanie języka naturalnego, uczenie maszynowe, system korekty błędów gramatycznych, uczenie głębokie, głębokie sieci neuronowe, korekta języka polskiego

* Projekt „Technologia kontekstowego rozumienia języka pisanego na potrzeby poprawy błędów oraz automatycznej oceny zrozumiałości tekstu", prowadzony przez Ermlab Sp. z o.o. w latach 2019–2021, jest finansowany ze środków Narodowego Centrum Badań i Rozwoju, w ramach programu Szybka Ścieżka, na podstawie umowy nr POIR.01.01.01-00-1128/18-00.

Abstract
The aim of the study is to present a research and development project focused on the development of contextual understanding of the written language for the purpose of an automatic error correction system for the Polish language. The presented ideas are clearly part of the current research (mainly in the field of English linguistics) on the use of natural language processing and machine learning for the development of GEC (grammatical error correction) systems. The first section of the article discusses the current state of research in a problem-oriented manner, i.e. the most important concepts in the field of creating GEC systems. The next paragraphs present the aim of the research and a brief methodological introduction. The penultimate paragraph points out the most important issues related to the development of the research corpus.

Keywords: natural language processing, machine learning, grammatical error correction system, deep learning, deep neutral networks, proofreading of Polish language

1. Automatyczna poprawa błędów językowych języka naturalnego – przegląd metod i dostępnych narzędzi

W ostatnim czasie problem tworzenia systemów do automatycznej poprawy błędów językowych języka naturalnego (tj. systemów GEC – *grammatical error correction*) jest kwestią często podejmowaną – zarówno na gruncie teoretycznym, jak i praktycznym. Warto jednak już na tym etapie rozważań wspomnieć, że zdecydowana większość tego rodzaju analiz dotyczy angielszczyzny lub to właśnie na gruncie angielszczyzny wykazuje najwyższą skuteczność (co jest zrozumiałe, zwłaszcza jeśliby wziąć pod uwagę rozpowszechnienie tego języka, a także jego specyfikę, tj. fakt, że jest to system pozycyjny/amorficzny, a więc pozbawiony rozbudowanej fleksji). Prace dotyczące języka polskiego, przede wszystkim te praktyczne, stanowią natomiast wciąż niedopracowany obszar dociekań (o czym poniżej).

U zarania prace nad powstaniem systemów GEC prowadzono głównie z wykorzystaniem metod słownikowych lub statystycznych. Budując narzędzia, wykorzystywano zatem np. algorytmy, na podstawie których obliczano tzw. odległości edycyjne – np. odległość Levenshteina[1]/Levenshteina–Damerau (zob. Levenshtein 1966; Damerau 1964; Kerninghan 1990) bądź odległość Jaro/Jaro–Winklera (zob. Jaro 1989, 1995; Winkler 1990, 2006) – czyli miary odmienności dwu tekstów. Z praktycznego punktu widzenia prace te były dość proste – sprowadzały się do sporządzenia metryk zdających sprawę z najmniejszych liczb działań prostych (takich jak: zamiana, usunięcie lub

[1] Uważa się, że metoda ta stanowi uogólnienie tzw. odległości Hamminga (por. Hamming 1950).

dodanie znaków lub przestawienie dwu sąsiadujących ze sobą znaków) koniecznych do przekształcenia jednego skończonego ciągu znaków w drugi. Choć metody te dobrze sprawdzają się np. w systemach antyplagiatowych i prostych narzędziach do weryfikacji poprawności zapisu, to zastosowane w celu zaawansowanej automatycznej korekty błędów języka polskiego, nie dają satysfakcjonujących wyników (z uwagi chociażby na zaburzone odległości edycyjne w wypadku analizy wyrazów składających się ortograficznie z dwu- lub trójznaków).

Wśród modeli statystycznych popularnością cieszyły się z kolei np. te oparte na wykorzystywanej z powodzeniem w procesach rozpoznawania mowy metodzie n-gramów (najczęściej: bi- lub trigramów). Przy pomocy tego podejścia można wyliczyć podobieństwo między określonymi ciągami znaków na podstawie liczby wspólnych podciągów; każdorazowo potrzeba do tego odpowiedniej wielkości korpusu, którego analiza pozwala na stworzenie modelu n-gramowego zdającego sprawę m.in. z liczby wystąpień w datasecie sekwencji o określonej długości (tj. o długości *n*) (zob. Jurafsky, Martin 2009; Ziółko, Skurzok 2011). Choć uważa się, że metoda ta jest dość prosta i łatwo skalowalna, to stworzenie przy jej pomocy rozwiązania o satysfakcjonujących możliwościach predykcyjnych (czyli dającego dobre wyniki dla tekstów niewystępujących w korpusie) jest trudne do osiągnięcia, ponieważ wymaga dostarczenia i przetworzenia ogromnej ilości danych językowych. Opracowywane na tej podstawie modele są bowiem w stanie sprawnie operować tylko tymi danymi, które wystąpiły w korpusie – nie „douczają się", tj. jeśli dane słowo lub sekwencja nie pojawiły się w datasecie, model ich nie rozpozna.

Podejmowane w ciągu kilku minionych lat próby opracowania systemów GEC dążą zatem do tego, by zaprojektować rozwiązanie o ponadprzeciętnych możliwościach w zakresie rozpoznawania błędów; idzie tu o odnalezienie zależności pomiędzy danymi a „etykietami" (np. wskazaniami typu błędu, sposobu jego poprawy lub wersjami po korekcie) po to, by dla możliwie jak najbardziej zróżnicowanych danych móc przewidzieć poprawną etykietę. W odniesieniu do korekty błędów sprowadza się to do tego, by na podstawie analizy danych korpusowych model potrafił rozpoznać cechy danego języka, a w konsekwencji mógł wskazywać dużą liczbę zapisów nieprawidłowych przy jednoczesnym sugerowaniu trafnych poprawek. Tego typu rozwiązania tworzy się obecnie przy wykorzystaniu metod uczenia maszynowego (*machine learning*), a konkretnie – uczenia głębokiego (*deep learning*) oraz głębokich sieci neuronowych (*deep neutral networks*). W zależności m.in. od architektury sieci i tego, jak przebiega proces „uczenia się", wyróżnia

się kilka podejść. W artykule zostanie wskazane najpopularniejsze z nich, tj. to oparte na sieciach neuronowych typu encoder-decoder.

Wspomniane sieci neuronowe wykorzystuje się z powodzeniem w tzw. tłumaczeniu maszynowym (*machine translation*). Upraszczając, sposób ich działania sprowadza się do „zakodowania" informacji (za to odpowiada encoder, pobierając sekwencje wyjściowe, przetwarzając je i gromadząc informacje dla każdego z elementów w postaci wektorów) oraz odpowiedniego ich „odkodowania" (do tego służy decoder). Tak skonstruowany algorytm uczy się na podstawie danych wektorowych wykorzystywanych w fazie generowania odpowiedzi przez decoder; po właściwym przetrenowaniu modelu (tj. wtedy, gdy zakończy się faza uczenia i testowania) decoder nie tylko jest w stanie biegle operować danymi pochodzącymi z encodera, lecz także potrafi dokonywać trafnych predykcji (zob. np. Junczys-Dowmunt, Grundkiewicz 2016). Tak zaprojektowany model GEC będzie zatem potrafił wskazywać zapisy niepoprawne oraz sugerować sposób ich korekty, czyli np. rozpozna błąd składniowy w zdaniu: **Tomek poszedł do sklep* i zaproponuje poprawną wersję (*Tomek poszedł do sklepu*).

Efektem praktycznego zainteresowania zagadnieniami GEC są natomiast funkcjonujące na rynku różnego typu aplikacje, które wśród swoich funkcjonalności wyróżniają (wyłącznie lub między innymi) wskazywanie zapisów błędnych oraz sugerowanie prawidłowych. Dominują tu rozwiązania opracowane z myślą o języku angielskim, takie jak np. Grammarly, LanguageTool, Ginger Software, Hemingway Editor, ProWritingAid, Scribens, GrammarCheck czy Zoho.

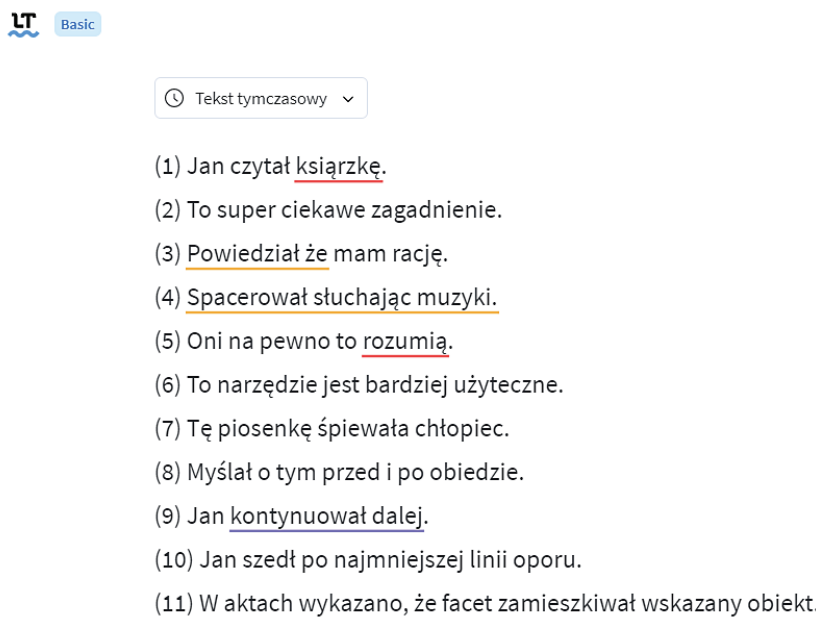
Co ciekawe, pomimo niewielu prac naukowych dotyczących tworzenia systemów GEC dla polszczyzny[2], lista narzędzi automatycznej korekty tekstów w języku polskim jest całkiem obszerna. Wymienić tu bowiem można aż trzy aplikacje: LanguageTool, iKorektor oraz KorektorTekstu; moduł do korekty błędów dołączany jest ponadto do najpopularniejszych programów edycyjnych (np. Microsofot Word, OpenOffice i Google Docs). Warto przyjrzeć się bliżej temu, jaką skuteczność poprawy językowych uchybień można przy ich pomocy osiągnąć, tj. jakie typy błędów są przez te narzędzia rozpoznawane. Wykorzystajmy do tego minizbiór zdań testowych reprezentujących każdy z rodzajów błędów językowych (Markowski 2005):

(1) **Jan czytał ksiąrzkę.*
(2) **To super ciekawe zagadnienie.*
(3) **Powiedział że mam rację.*

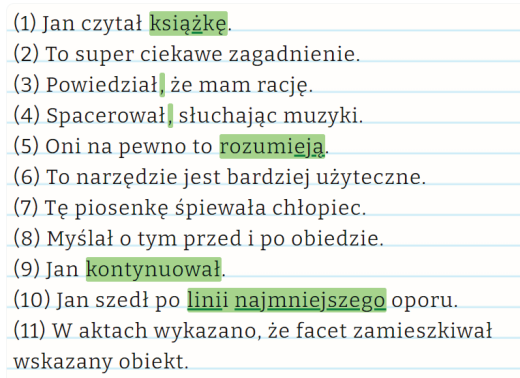
[2] Zob. Miłkowski 2008, 2010.

(4) **Spacerował słuchając muzyki.*
(5) **Oni na pewno to rozumią.*
(6) **To narzędzie jest bardziej użyteczne.*
(7) **Tę piosenkę śpiewała chłopiec.*
(8) **Myślał o tym przed i po obiedzie.*
(9) **Jan kontynuował dalej.*
(10) **Jan szedł po najmniejszej linii oporu.*
(11) **W aktach wykazano, że facet zamieszkiwał wskazany obiekt.*

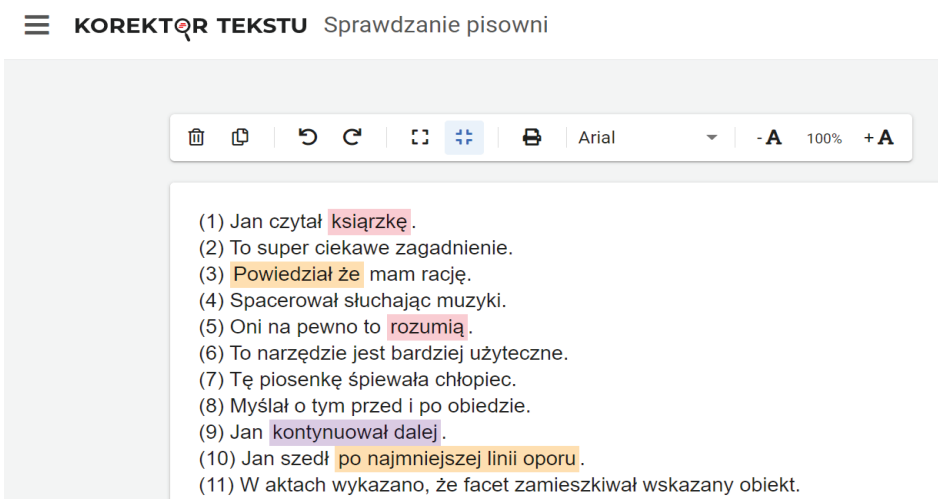
Zdania (1)–(2) powinny zostać skorygowane pod względem ortograficznym (*książkę*, *superciekawe*), wypowiedzi (3)–(4) wymagają poprawek interpunkcyjnych (*Powiedział, że…*; *Spacerował, słuchając…*), w realizacjach (5)–(6) należy zasugerować prawidłowe formy fleksyjne (*rozumieją*, *użyteczniejsze*), przykłady (7)–(8) zawierają błędy składniowe (dotyczące związku głównego – *śpiewał chłopiec* oraz skrótów składniowych – *przed obiadem i po obiedzie*), zdania (9)–(10) – usterki leksykalne (*kontynuował*, *szedł po linii najmniejszego oporu*), a wypowiedź (11) trzeba by poprawić pod kątem stylistycznym (*mężczyzna zamieszkiwał*).



Ryc. 1. Korekta tekstu w aplikacji LanguageTool



Ryc. 2. Korekta tekstu w aplikacji iKorektor



Ryc. 3. Korekta tekstu w aplikacji KorektorTekstu

Analiza powyższych przykładów pozwala stwierdzić, że aplikacje te różni rodzaj i zakres wskazywanych błędów. Znakomita większość z nich bardzo dobrze radzi sobie z rozpoznawaniem rażących usterek ortograficznych, zob. (1), fleksyjnych, zob. (5), i leksykalnych, zob. (9)–(10). Zadowalające wyniki osiągają one również w korekcie interpunkcyjnej prostych konstrukcji składniowych, zob. (3). Należy jednak zauważyć, że narzędzia te nie znajdują zastosowania w wypadku bardziej zaawansowanych błędów ortograficznych, zob. (2), fleksyjnych, zob. (6), oraz składniowych, zob. (8); co ciekawe, nie wskazują one także usterek składniowych w zakresie związku głównego, zob. (7). Wynika to zapewne z tego, że bazują one na systemach regułowych zbudowanych siłami społeczności lub na rozwiązaniach wykorzystujących

automaty skończone. Takie podejście nie sprawdza się w praktyce dla języka polskiego, tj. systemu silnie fleksyjnego, o zaawansowanej składni. Wydaje się zatem, że uzasadnione jest stworzenie lepszego modelu, który mógłby wypełnić tę lukę.

2. Cel projektu

Głównym celem prezentowanego projektu jest opracowanie narzędzia, które dokonywałoby automatycznej, kompleksowej korekty językowej tekstów języka polskiego, tj. wskazywałoby zapisy niepoprawne i podpowiadało prawidłowe w zakresie każdej z kategorii błędów językowych w ujęciu Andrzeja Markowskiego (2005). Prace badawcze prowadzone są w latach 2019–2021, a zaprojektowane modele zostaną zaimplementowane w postaci dodatku (wtyczki) do przeglądarek Google Chrome i Mozilla Firefox; będą one działać również w ramach systemu zarządzania treścią Wordpress oraz w aplikacji Microsoft Office 365.

Charakteryzowane badania są w oczywisty sposób związane z zagadnieniem przetwarzania języka naturalnego oraz ze zalgorytmizowanym opisem polszczyzny, zatem pod wieloma względami wpisują się w rozwój i potrzeby współczesnej lingwistyki komputerowej. Opisywane w punkcie 1. metody projektowania systemów GEC były dotychczas rozwijane przede wszystkim na gruncie językoznawstwa angielskiego i to właśnie w tym języku wykazują wysoką skuteczność. Ich zastosowanie w odniesieniu do polszczyzny wiąże się z oczywistą koniecznością zmierzenia się ze specyfiką fleksyjno-składniową tego systemu językowego. Prowadzone badania będą zatem stanowić też swego rodzaju weryfikację opracowanych podejść, a co za tym idzie również ich modyfikację i rozwinięcie.

3. Metodologia badań

Aplikacja będąca przedmiotem opisu w niniejszym artykule oparta będzie na modelu opartym na metodach głębokiego uczenia, z wykorzystaniem głębokich sieci neuronowych o architekturze typu transformer[3] pracujących na zasadzie encoder-decoder.

Jak zostało już zasygnalizowane (por. p. 1), w wypadku rozwiązań projektowanych z wykorzystaniem metod uczenia maszynowego kluczowym czynnikiem jest wielkość korpusu treningowego – im więcej danych model

[3] Zob. Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, Polosukhin 2017.

zanalizował, tym wyższa jest jego skuteczność. Zbudowanie właściwego korpusu (tj. odpowiednio obszernego i zróżnicowanego) będzie zatem stanowić jeden z ważniejszych etapów planowanych prac badawczych. Zakładany sposób gromadzenia danych językowych wynika bezpośrednio z przyjętych metod budowania modelu. Uczenie algorytmów stanowiących bazę dla aplikacji będzie miało bowiem charakter nadzorowany. Istota tego procesu wymaga opracowania dwu korpusów, w skład których wchodzić będą pary zdań (każdorazowo rzecz będzie sprowadzać się do podania zdania niepoprawnego zawierającego ściśle scharakteryzowany błąd językowy i zdania poprawnego, tj. takiego, w którym przeprowadzona została korekta określonego błędu). Upraszczając, istotą procesu uczenia się algorytmów będzie porównywanie wygenerowanych zdań skorygowanych przez sieć ze zdaniami poprawnymi zgromadzonymi w korpusie.

Ponieważ zdania błędne muszą być zarówno odpowiednio poprawione, jak i właściwie opisane (co wymagałoby niezwykle czasochłonnej ingerencji człowieka), zdecydowano się na wykorzystanie metod augmentacji danych, tj. na wygenerowanie sztucznych danych leksykalnych. W praktyce sprowadzać się to będzie do zgromadzenia zdań poprawnych (pochodzących np. z literatury, polskiej Wikipedii lub innych redagowanych językowo źródeł internetowych) i opracowania reguł, na podstawie których będzie można zaimplementować do tych wypowiedzi błędy językowe. Niezwykle pomocna w tym procesie będzie również analiza morfosyntaktyczna; tylko w ten sposób da się przeprowadzić odpowiednią segmentację wyrazową (tokenizację) zdań (traktowanie wyrazów w sposób fleksyjny, nie: ortograficzny). Do jej przeprowadzenia planuje się wykorzystać analizator morfologiczny Morfeusz.

System automatycznej poprawy błędów dla języka polskiego, mający stanowić efekt prac badawczych prowadzonych w ramach omawianego projektu, będzie korzystał zarówno z dokonań uczenia maszynowego, jak i z dorobku teoretycznego w zakresie normatywnego opisu współczesnej polszczyzny. Można przypuszczać, że prowadzone analizy przyczynią się również do rozwoju badań nad automatycznym przetwarzaniem polszczyzny oraz będą znaczącym uzupełnieniem prac zmierzających do zaprojektowania skutecznego systemu GEC dla języka polskiego.

Źródła

Ginger Software, <https://www.gingersoftware.com/>, dostęp: 25.03.2021.
GrammarCheck, <https://www.grammarcheck.net/editor/>, dostęp: 25.03.2021.
Grammarly, <https://www.grammarly.com/>, dostęp: 25.03.2021.
Hemingway Editor, <https://hemingwayapp.com/>, dostęp: 25.03.2021.
iKorektor, <https://ikorektor.pl/>, dostęp: 25.03.2021.

KorektorTekstu, <https://www.korektortekstu.pl/>, dostęp: 25.03.2021.
LanguageTool, <https://languagetool.org/pl>, dostęp: 25.03.2021.
ProWritingAid, <https://prowritingaid.com/>, dostęp: 25.03.2021.
Scribens, <https://www.scribens.com/>, dostęp: 25.03.2021.
Zoho, <https://www.zoho.com/writer/free-grammar-checker.html>, dostęp: 25.03.2021

Literatura

Bird S., Loper E., Klein E. (2009): *Natural Language Proceesing with Python*. USA.

Damerau F. J. (1964): *A technique for computer detection and correction of spelling errors*. „Communications of the ACM". T. 7, nr 3, s. 171–176.

Evert S., Krenn B. (2005): *Using small random samples for the manual evaluation of statistical association measures*. „Computer Speech & Language" nr 19 (4), s. 450–466.

Hamming, R. W. (1950): *Error detecting and error correcting codes*. „The Bell System Technical Journal" nr 29 (2), s. 147–160.

Jaro M. A. (1989): *Advances in record linkage methodology as applied to the 1985 census of Tampa Florida*. „Journal of the American Statistical Association" nr 84 (406), s. 414–420.

Jaro M. A. (1995): *Probabilistic linkage of large public health data file*. „Statistics in Medicine" nr 14 (5–7), s. 491–498.

Junczys-Dowmunt M., Grundkiewicz R. (2016): *Phrase-based Machine Translation is State-of-the-Art for Automatic Grammatical Error Correction*. [W:] *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics*. Austin, s. 1546–1556.

Jurafsky D., Martin J. H. (2009): *Speech and Language Processing*, <https://web.stanford.edu/~jurafsky/slp3/>, dostęp: 25.03.2021.

Kerninghan M. D., Church K. W., Gale W. A. (1990): *A spelling correction program based on a noisy channel model*. „Proceedings of the COLING-90", s. 205–211.

Levenshtein V. I. (1966): *Binary codes capable of correcting deletions, insertions, and reversals*. „Soviet Physics Doklady" nr 10 (8), s. 707–710.

Markowski A. (2005): *Kultura języka polskiego. Teoria. Zagadnienia leksykalne*. Warszawa.

Miłkowski M. (2008): *Automated Building of Error Corpora of Polish*. [W:] *Corpus Linguistics, Computer Tools, and Applications – State of the Art*. Red. B. Lewandowska-Tomaszczyk, s. 631–639.

Miłkowski M. (2010): *Developing an open-source, rule-based proofreading tool*. „Software – Practice and Experience" nr 40 (7), s. 543–566.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., Polosukhin I. (2017): *Attention Is All You Need*. [W:] *31st Conference on Neural Information Processing Systems*. USA, s. 1–15.

Winkler W. E. (1990): *String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage*. „Proceedings of the Section on Survey Research Methods, American Statistical Association", s. 354–359.

Winkler W. E. (2006): *Overview of Record Linkage and Current Research Directions*. Waszyngton.

Ziółko B., Skurzok D. (2011): *N-grams model for Polish*. „Speech and Language Technologies". T. 2, s. 107–126.