JCSI 21 (2021) 316–323
Received: 09 September 2021
Accepted: 20 September 2021

Examination of text's lexis using a Polish dictionary

Badanie leksyki tekstu na podstawie słownika języka polskiego

Roman Voitovych*, Edyta Łukasik

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

This paper presents an approach to compare and classify books written in the Polish language by comparing their lexis fields. Books can be classified by their features, such as literature type, literary genre, style, author, etc. Using a preassembled dictionary and Jaccard index, the authors managed to prove a lexical likeness for books. Further analysis with the PAM clustering algorithm presented a lexical connection between books of the same type or author. Analysis of values of similarities of any particular field on one side and some anomalous tendencies in other cases suggest that recognition of other features is possible. The method presented in this article allows to draw conclusions about the connection between any arbitrary books based solely on their vocabulary.

Keywords: natural language processing; lexis analysis; Jaccard similarity coefficient; Partitioning Around Medoids;

Streszczenie

Artykuł prezentuje metodę porównania i klasyfikacji książek napisanych w języku polskim na podstawie ich leksyki. Książki można dzielić, korzystając z ich cech, np. rodzaju literatury, gatunku literackiego, stylu, autora itp. Korzystając ze skompilowanego słownika i indeksu Jaccarda, udowodniona została hipoteza dotycząca podobieństwa książek rozpatrywanego pod kątem ich leksyki. Kolejna analiza za pomocą algorytmu klastrowego PAM wskazuje na związek leksykalny pomiędzy książkami jednego rodzaju literatury lub autora. Analiza wartości współczynników poszczególnych obszarów z jednej strony i anomalia w zachowaniu w niektórych przypadkach sugeruje, że wyodrębnienie kolejnych cech jest możliwe. Metoda przedstawiona w tym artykule pozwala wyciągać wnioski o relacjach między książkami, korzystając wyłącznie z ich słownictwa.

Słowa kluczowe: przetwarzanie języka naturalnego; analiza leksyczna; indeks Jaccarda; Partitioning Around Medoids;

*Corresponding author

Email address: voytroman@protonmail.com (R. Voitovych)

1. Introduction

The purpose of this article is to investigate the possibility to automatically compare different books with each other by their lexis fields and draw conclusions regarding their features, such as literature type, literary genre, scientific field, author, etc. The hypothesis concerning connection between lexis likeness and common characteristics is studied. The target language of this study is Polish, which has an enormous amount of morphology-driven declensions which causes much greater difficulties for any natural language processing tool. For example, English and other, more inflectional, Germanic and Romance languages are much more simpler than Polish from this point of view.

This idea implies the usage of a thorough dictionary including every possible declension, that allows to lemmatise texts and analyse them as a set of lemmas. It also requires a storage solution, both for the dictionary and analysis results, a comparison algorithm such as a similarity or difference coefficient of measures and finally a clustering algorithm to ease the analysis of big comparison scenarios.

Former works [1-5], some of which examine similar literature classification problem [6-7], present a lot of possible solutions for text differentiation yet part of them concerns not only lexical similarity but also a semantic one which is not in the scope of this work. The semantic comparison allows to get a match between two texts by their meanings without relying on the words used. Yet long texts like novels, textbooks or legal codes have a tendency to contain an enormous amount of ideas, events and plots and that may render semantic analysis's results questionable. Other algorithms group neighbouring words into structures like vectors but this research treats every word independently. The primary idea behind this research is that the choice of words in any book is defined by their features.

The theme of this thesis falls into a domain of general natural language processing [8], which in turn is a comprehension approach of computational linguistics [9]. Some elements of language morphology studies concerning lemmas and declensions from [10] are used. Also, the Jaccard similarity coefficient [11] and Partitioning Around Medoids implementation of the k-medoids algorithm [12] are used as research tools.

Three primary research questions of this article are:

Q1. Does the comparison between lexis fields allow to state the overall difference in terms of any grouping features?

Q2. Is the lexical similarity of books augmented by the mutual author?

Q3. Is there a significant difference between different types of non-fiction and fiction literature?

2. Materials and methods

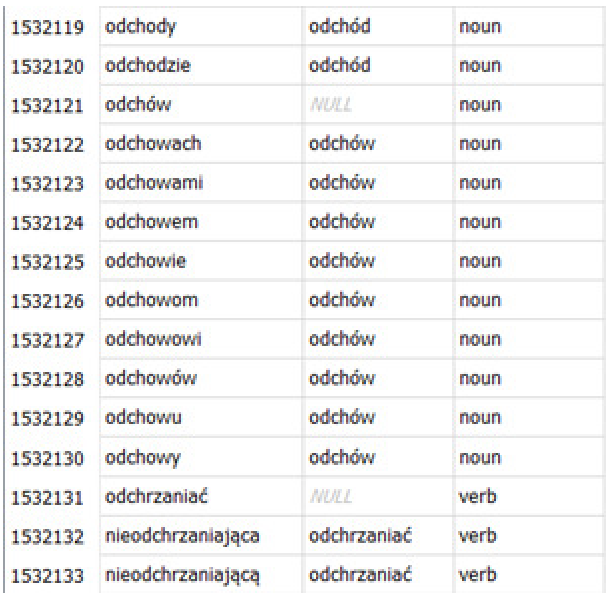
2.1. Dictionary

The SJP (Słownik Języka Polskiego) is a free internet database devoted to tracking every possible Polish word and its morphology [13]. It provides a quite reliable index of Polish words including their declensions. This dictionary appears to have an extremely high word coverage ranging from 97% to 99% for average fiction book's vocabulary but its form is not fully compatible with chosen database schema. Firstly, every word in the database is unique in the scope of one table, thus meaning that the number of declensions will be reduced to only one instance. This rule does not exclude any possible repetition of lemmas as declensions of other lemmas. In such cases, all their declension forms will be discarded with only the lemma form left. Without this procedure, there will be exactly 2 388 or 1.13% of all lemmas stored in the database as declensions.

Many of the words stored in this database are not common names but proper names, abbreviations, initialisms, phrases, etc. Importing function separates proper names (recognised by initial capital letters) into table "proper" and initialisms with abbreviations (recognised by non-initial capital letters, dots and hyphens) into table "abbreviation" discarding all words containing spaces and other non-letter symbols. That process allows to get a refined collection of common words stored in the table "common", yet it does not exclude words derived from proper nouns, foreign words and or similar odd cases.

The newly imported SQLite lexis database with all aforementioned corrections applied has 172 705 lemmas from which 165 737 are mentioned as lemmas in declension entries. The absolute number of all common word entries is 3 980 250. It also includes 303 286 proper word inflexions and 11 241 abbreviation and initialism forms. Its fragment is shown in Table 1.

Table 1: A sample slice of common word forms from the lexis database.



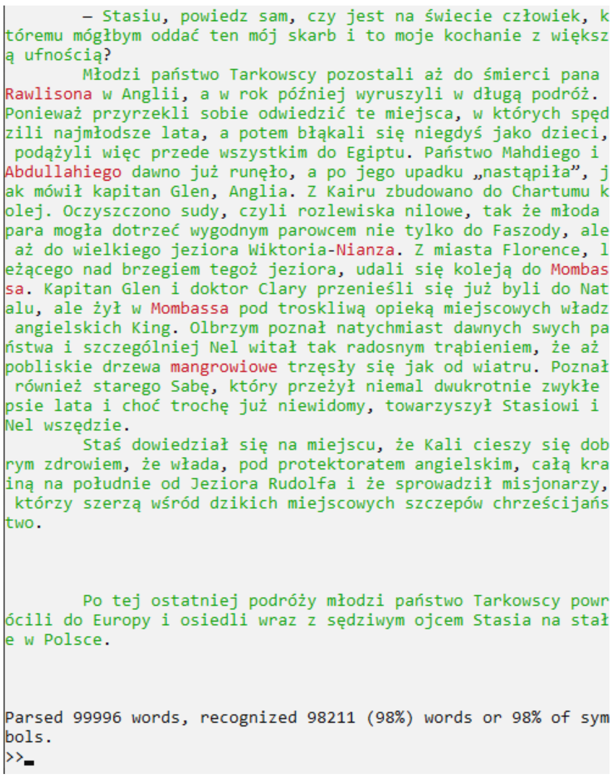
1532119	odchody	odchód	noun
1532120	odchodzie	odchód	noun
1532121	odchów	NULL	noun
1532122	odchowach	odchów	noun
1532123	odchowami	odchów	noun
1532124	odchowem	odchów	noun
1532125	odchowie	odchów	noun
1532126	odchowom	odchów	noun
1532127	odchowowi	odchów	noun
1532128	odchowów	odchów	noun
1532129	odchowu	odchów	noun
1532130	odchowy	odchów	noun
1532131	odchrzaniać	NULL	verb
1532132	nieodchrzaniająca	odchrzaniać	verb
1532133	nieodchrzaniającą	odchrzaniać	verb

2.2. Parser

Research purposes require a robust parser that can read any text written in the Polish language and map every recognised word to an individually created database file based on the lexis database schema. It has additional columns named "count" and "probability", which allow to track the number of parsed words. Such an approach allows to keep track of every word's properties without the need to store the entire dictionary.

The first step is to read text files containing target text. Popular text file formats can store all Polish letters in several encodings like Windows-1250 or Unicode, but the availability of many foreign words from French, Spanish, German and other languages which use Latin diacritics forces the use of variable-width encoding. The chosen code page is the widespread UTF-8.

All of the tools in this article are written in Windows 10 environment. The chosen implementation language is C due to its robust speed and efficiency. More specifically it uses the MinGW-w64 compiler. This combination already causes problems concerning UTF-8 parsing because of default wchar.h library appears to have conflict with a console window that misreads Unicode input. Secondly, all of the conversion and reading functions do not work with symbols that have multiple-byte width. The first problem is omitted by using the Windows-1250 code table at input while all needed read and conversion functions are manually restored.



```
        – Stasiu, powiedz sam, czy jest na świecie człowiek, k
tóremu mógłbym oddać ten mój skarb i to moje kochanie z większ
ą ufnością?
        Młodzi państwo Tarkowscy pozostali aż do śmierci pana
Rawlisona w Anglii, a w rok później wyruszyli w długą podróż.
Ponieważ przyrzekli sobie odwiedzić te miejsca, w których spęd
zili najmłodsze lata, a potem błąkali się niegdyś jako dzieci,
 podążyli więc przede wszystkim do Egiptu. Państwo Mahdiego i
Abdullahiego dawno już runęło, a po jego upadku „nastąpiła”, j
ak mówił kapitan Glen, Anglia. Z Kairu zbudowano do Chartumu k
olej. Oczyszczono sudy, czyli rozlewiska nilowe, tak że młoda
para mogła dotrzeć wygodnym parowcem nie tylko do Faszody, ale
 aż do wielkiego jeziora Wiktoria-Nianza. Z miasta Florence, l
eżącego nad brzegiem tegoż jeziora, udali się koleją do Mombas
sa. Kapitan Glen i doktor Clary przenieśli się już byli do Nat
alu, ale żył w Mombassa pod troskliwą opieką miejscowych władz
 angielskich King. Olbrzym poznał natychmiast dawnych swych pa
ństwa i szczególniej Nel witał tak radosnym trąbieniem, że aż
pobliskie drzewa mangrowiowe trzęsły się jak od wiatru. Poznał
 również starego Sabę, który przeżył niemal dwukrotnie zwykłe
psie lata i choć trochę już niewidomy, towarzyszył Stasiowi i
Nel wszędzie.
        Staś dowiedział się na miejscu, że Kali cieszy się dob
rym zdrowiem, że włada, pod protektoratem angielskim, całą kra
iną na południe od Jeziora Rudolfa i że sprowadził misjonarzy,
 którzy szerzą wśród dzikich miejscowych szczepów chrześcijańs
two.



        Po tej ostatniej podróży młodzi państwo Tarkowscy powr
ócili do Europy i osiedli wraz z sędziwym ojcem Stasia na stał
e w Polsce.



Parsed 99996 words, recognized 98211 (98%) words or 98% of sym
bols.
>>▬
```

Figure 1: Output produced by parser reading "W pustyni i w puszczy".

While the parser reads UTF-8 text file byte by byte it arranges it in words, counts them and tries to find a match in the lexis database. To find the match it rotates the letter case of initial and other letters and breaks down words with hyphens or dots into smaller parts which, for example, allows to analyse hyphenated compound adjectives as separate words. When a match is found its corresponding entry is added to the text's database file. When the parser recognises a repeated use of a previously added word it simply amplifies the "count" value by one.

After reading the whole file program writes the total number of recognised and unrecognised words and symbols into a separate "stats" table. It allows to normalise results by dividing count number by total number and to store this value in the "probability" field.

During this process, the console window presents the progress in the form of a plain currently parsed coloured text as it is presented in Figure 1, where green colour indicates recognised symbols, red – unrecognised and black – ignored ones.

2.3. Analyser

The current state of science presents many different methods to compare two sets of labelled data [3,5] part of which are purely lexical and others employ a semantical analysis, yet only lexical ones are used here. To show off the basic idea of text's attributes strictly defined by its vocabulary the most basic and widely known Jaccard similarity coefficient [5,11] is used:

$$\mathrm{J}(\mathrm{A},\mathrm{B}) = \frac{|\mathrm{A}\cap \mathrm{B}|}{|\mathrm{A}| + |\mathrm{B}| - |\mathrm{A}\cap \mathrm{B}|} \tag{1}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are finite sets. The definition implies that the Jaccard index receives values between and including 0 and 1, where 0 means no similarity between sets and 1 shows that $\boldsymbol{A}$ and $\boldsymbol{B}$ are identical.

Listing 1: Jaccard index implementation.

```
    sqlite3_exec(_compare_db, SQL_JACCARD_INNER,
                 jaccard_index_inner_callback, NULL, NULL);
    sqlite3_exec(_compare_db, SQL_JACCARD_LEFT,
                 jaccard_index_left_callback, NULL, NULL);
    sqlite3_exec(_compare_db, SQL_JACCARD_RIGHT,
                 jaccard_index_right_callback, NULL, NULL);

    result = _jaccard_ab/(_jaccard_a+_jaccard_b-_jaccard_ab);
    return result;
}

int jaccard_index_inner_callback(void *null_data, int argc,
                                  char **argv, char **argcol){
    double a, b;
    a = atof(argv[0]);
    b = atof(argv[1]);
    if(a>b){
        _jaccard_ab+=b;
        _jaccard_a+=a-b;
    }
    else{
        _jaccard_ab+=a;
        _jaccard_b+=b-a;
    }
    return 0;
}
```

The implementation of this formula (Listing 1) needs to take into consideration the schema of statistical data. Every text database contains a list of words and the probability of their occurrence. Before any comparison can be made analyser module converts declensions into lemmas summarising their probability measures and writes to a comparison database in two separate tables from two texts' databases correspondingly. $\mathbf{A}\cap\mathbf{B}$ is defined as a lesser of two probabilities for every match between lemmas. The remainder of those operations is added to the rest of unmatched lemmas either in set $\boldsymbol{A}$ or $\boldsymbol{B}$.

2.4. Partitioning Around Medoids algorithm

Using the aforementioned implementation of the Jaccard index program compares n books between each other and builds $n \times n$ symmetrical matrix from which every pair value (a, b) hold a coefficient between books a and b. Yet it is still hard to draw conclusions concerning classification without the appropriate clustering algorithm.

The metric in which those similarity indices are measured does not use the standard Euclidean geometry which is required for a k-means clustering algorithm to compute centroids. The only measures available are similarities which do not allow to derive any dependent value without the metric. But similar k-medoids algorithm solves this problem by using only existing points as centroids and requires the set of distances or in this case inverted similarities to run properly.

In this paper a Partitioning Around Medoids (PAM) algorithm [12] which uses a greedy search to speed up the process was implemented. It aims to find clusters with high degree of similarity while the similarity between clusters stays low. There is no strict rule for selecting the number of clusters k so it must be manually determined for every scenario.

PAM has two distinct phases – BUILD and SWAP. During the BUILD phase algorithm selects initial predefined number of centroids. Then in the SWAP phase, it considers pairs of objects with an object selected as centroids and an unselected object to swap them if it decreases the total dissimilarities inside clusters. Every time the algorithm changes an array of selected objects it recalculates two values for every object p - D_p, the dissimilarity between p and the closest selected object and E_p, the dissimilarity to the second closest selected object. $d(i, j)$ is a dissimilarity between objects i and j.

The BUILD phase is initialised by adding a first object for which the sum of distances to other objects is minimal. Then it considers unselected objects to select by computing and choosing the best total gain for every pair of unselected objects i and j:

$$g_i = \sum C_{ji} \tag{2}$$

where

$$C_{ji} = \max\{D_j - d(j, i), 0\} \tag{3}$$

It repeats this calculation $k - 1$ times resulting in a set of k centroids.

Then during the SWAP phase, it considers every pair of (i, h) where i is a selected object and h is an unselected object and calculates the contribution:

$$T_{ih} = \sum K_{jih} \tag{4}$$

where K_{jih} is the contribution of swapping object i and h concerning other unselected object j. If $d(j,i) > D_j$, then $K_{jih} = \min\{d(j,h) - D_j, 0\}$, but if $d(j,i) = D_j$, then $K_{jih} = \min\{d(j,h), E_j\} - D_j$. After selecting the minimal contribution $T_{ih} < 0$ algorithm swapes the corresponding pair (i,h), recalculates values D_p and E_p and tries again.

If total contribution T_{ih} equals zero or more, then the best combination is already found and PAM is stopped. The values of D_p determine to which cluster belongs object p.

Unfortunately due to the unknown metric, there is no way to visualise results from a plain list of clusters and their objects.

3. Research and selected results

Any if not all features, can be purely a subject of semantics and not lexis. Those two concepts are quite related, but lexical units can have an enormous number of semantic definitions and a lot of them are incredibly flexible. One could argue that semantic units can use only a limited number of lexical units unless the author is willing to communicate using words, which are detached from their original meanings. Even if it happens, it usually distorts the denotation of complete phases and still relies on dictionary definitions of separate words, thus making the word's connotations heavily dependent on its original meaning.

Considering that analysis used in this article is purely lexical and does not use semantics or relations between parsed words, such as vectors, its proper function is fully reliant on the connection between the text's dictionary and its properties.

Before concluding any further research the proof using the comparison between loosely related and absolutely similar pairs of books in terms of features is needed. If this relation and comparison by Jaccard coefficient are correlating, then a further k-medoids clustering allows us to group books into clusters with some common features.

All of the scenarios mentioned below use Polish fiction and non-fiction medium-sized books as the test subjects. Fiction books are sourced entirely from the public domain and non-fiction books are taken either from Lublin University of Technology Publishing House or Journal of Laws of the Republic of Poland.

3.1. Theoretical constraints and implications

Research results require a definition of how lexis field similarity is represented by the Jaccard similarity coefficient works. Absolute results i.e. zero and one correspond to absent and total similarity accordingly, yet this is irrelevant in practical comparison. Those two values show one major metric difference between the two ideal states. In a case when a pair with one definite book on the left and one arbitrary book on the right it gets a coefficient value equal to one only when both lexis fields are identical. But to get zero it can use an indefinite number of different lexis fields for the second book all of which will have an unconstrained range of similarity between themselves.

Thus if $s(a,b) = 1$ and $s(a,c) = 1$ (where s is a function of similarity coefficient and sets a, b, c are dictionaries with elements which represent words) are true then the statement $s(b,c) = 1$ is also true, which also means that sets a, b, c are identical. But if $s(a,b) = 0$ and $s(a,c) = 0$ are true then the value of $s(b,c)$ stays unknown and can have any value between 0 and 1.

Also, those two states are probably unachievable in any real scenario. It is highly unlikely for long texts such as books to have an identical set of words with alike probabilities of their individual use, even if the contents of those books are very similar. The abundance of so-called function words such as conjunctions, prepositions, pronouns, etc. and the overall tendency to use high-frequency words makes zero and neighbouring values of coefficient very unlikely. Those two constraints mean that practical absolute values are distanced from their theoretical counterparts.

During the examination of one or more types of features it has to be assured that inside every chosen group of books there is a unique feature-based connection that is not present neither between books from different groups nor inside any other group. Secondly, there is a problem with feature selection. No characteristic is unique only to one element of one feature type. It is safe to assume that most of the literary divisions are intertwined with one another. Authorship, time of writing, publishing house, type of literature, literary style, genre and an unknown number of other divisions are probably connected with different but partially similar lexes. It means that the part of the resulting book's lexis is not concatenated from feature lexes but rather superimposed by their repeating parts, thus augmenting their frequency.

If the clustering algorithm successfully recreates initial grouping then there is surely a correlation between chosen feature likeness and the lexis likeness. However, it is essential to analyse how coefficients behave inside and outside the aforementioned groups to prove causation.

Consequently, two basic terms concerning mean similarities inside and outside these groups are introduced to draw conclusions regarding levels of similarity in those scenarios. *Background noise similarity* is a mean of all relations for every book pair outside the groups. In a properly build scenario it will correspond to a practical minimum for a chosen background feature field. Similarly, *common inner similarity* or separate *inner similarities* for every group show how much more similar those books are to the books of respective groups. The *foreground similarity* is the difference between *common inner similarity* and *background noise*. Its resulting value ignores the "noise" of unomittable basic vocabulary (function words and most frequent words) and common lexis of chosen background features. Plain terms *background* and *foreground* in this article represent sets of similarity indices outside the groups and inside the groups correspondingly.

Foreground similarity can be also used as a determiner for the most optimal number of clusters for a chosen scenario. Most clustering algorithms do not have any effective rules regarding the number of clusters and it is advised not to impose them but rather select the number of clusters manually. If the distinction between background noise and inner similarity is strict enough and their individual pair values do not deviate from means too much, then the PAM structuring by distances will work most efficiently with the right number of classes. Thus it may allow to automatically select the best number of clusters by choosing the division with the biggest resulting foreground similarity. If the supposed division number is not the optimal one, then there is an issue either with the feature's elements division or there is some other unknown stronger feature in play.

3.2. Scenario 1 – Absolute similarities

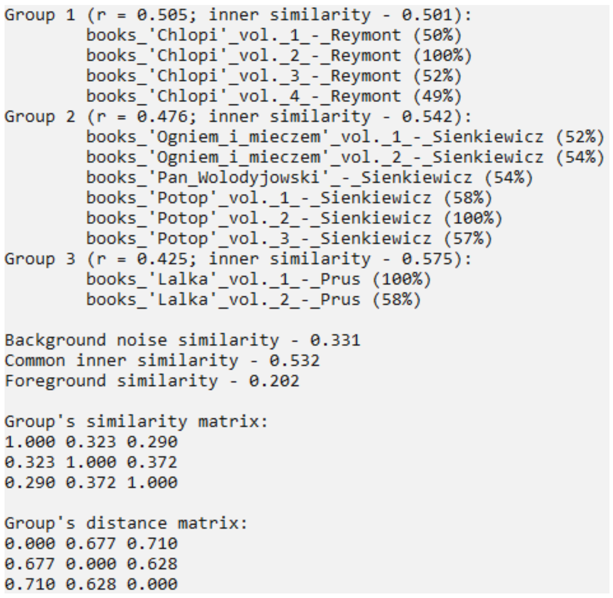
The first research question of this article asks whether the comparison between lexis fields allows spotting differences in their features at all. It is tested by comparing the similarity between mostly unrelated books with the similarity between exceptionally similar books. To do this the program chooses the best possible conditions to reach both practical absolute coefficient values.

To reach the practical maximum, books which have identical features have to be selected. While it seems to be impossible, there is a way to achieve this by breaking up one book into two or more parts. These parts will have the same characteristics as the whole book.

To reach the practical minimum value, a set of books with a minimal number of common features has to be selected. However, there is at least one common feature in this set of books – a mutual type of literature. Secondly, this feature may be much stronger than any other feature and thus it has to be examined separately. Due to an unknown number of probable divisions, it is very hard to assure that there are no other common characteristics between chosen books.

But even with the common type of literature and some other unknown features this scenario still has a small background feature similarity compared to the highest achievable foreground feature similarity. If the degree of lexical differences is greater or roughly equal between two parts of one book than to a completely unrelated book, then this method does not work, because it is probably the biggest possible feature distance between background and foreground.

Considering how publishers and authors break up their works into volumes and parts this scenario can use these two division methods and analyse every volume as a separate text. Books chosen are the two volumes of B. Prus's "Lalka" (99% and 98% coverage), four volumes of Reymont's "Chłopi" and all three entries in H. Sienkiewicz's Trilogy – "Ogniem_i_mieczem" (2 vol.), "Pan Wołodyjowski" (1 vol.) and "Potop" (3 vol.). All of these books are fiction books and have different styles, authors and etc.

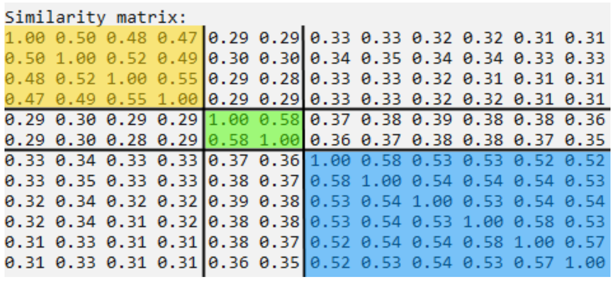

```
Group 1 (r = 0.505; inner similarity - 0.501):
        books_'Chlopi'_vol._1_-_Reymont (50%)
        books_'Chlopi'_vol._2_-_Reymont (100%)
        books_'Chlopi'_vol._3_-_Reymont (52%)
        books_'Chlopi'_vol._4_-_Reymont (49%)
Group 2 (r = 0.476; inner similarity - 0.542):
        books_'Ogniem_i_mieczem'_vol._1_-_Sienkiewicz (52%)
        books_'Ogniem_i_mieczem'_vol._2_-_Sienkiewicz (54%)
        books_'Pan_Wolodyjowski'_-_Sienkiewicz (54%)
        books_'Potop'_vol._1_-_Sienkiewicz (58%)
        books_'Potop'_vol._2_-_Sienkiewicz (100%)
        books_'Potop'_vol._3_-_Sienkiewicz (57%)
Group 3 (r = 0.425; inner similarity - 0.575):
        books_'Lalka'_vol._1_-_Prus (100%)
        books_'Lalka'_vol._2_-_Prus (58%)

Background noise similarity - 0.331
Common inner similarity - 0.532
Foreground similarity - 0.202

Group's similarity matrix:
1.000 0.323 0.290
0.323 1.000 0.372
0.290 0.372 1.000

Group's distance matrix:
0.000 0.677 0.710
0.677 0.000 0.628
0.710 0.628 0.000
```
Figure 2: The result of clustering 11 volumes taken from 2 books and one trilogy into 3 groups.

As shown in Figure 3, PAM structuring by Jaccard similarities acquired by dictionary comparison allows to successfully reassemble volumes into books and book series. Each cluster has a width between 0.42 and 0.51, while distances between those groups range between 0.63 and 0.71. Common inner similarity equals 0.53, while the background noise is 0.33, resulting in a 20% index difference between background and foreground.


```
Similarity matrix:
1.00 0.50 0.48 0.47|0.29 0.29|0.33 0.33 0.32 0.32 0.31 0.31
0.50 1.00 0.52 0.49|0.30 0.30|0.34 0.35 0.34 0.34 0.33 0.33
0.48 0.52 1.00 0.55|0.29 0.28|0.33 0.33 0.32 0.31 0.31 0.31
0.47 0.49 0.55 1.00|0.29 0.29|0.33 0.33 0.32 0.32 0.31 0.31
0.29 0.30 0.29 0.29|1.00 0.58|0.37 0.38 0.39 0.38 0.38 0.36
0.29 0.30 0.28 0.29|0.58 1.00|0.36 0.37 0.38 0.38 0.37 0.35
0.33 0.34 0.33 0.33|0.37 0.36|1.00 0.58 0.53 0.53 0.52 0.52
0.33 0.35 0.33 0.33|0.38 0.37|0.58 1.00 0.54 0.54 0.54 0.53
0.32 0.34 0.32 0.32|0.39 0.38|0.53 0.54 1.00 0.53 0.54 0.54
0.32 0.34 0.31 0.32|0.38 0.38|0.53 0.54 0.53 1.00 0.58 0.53
0.31 0.33 0.31 0.31|0.38 0.37|0.52 0.54 0.54 0.58 1.00 0.57
0.31 0.33 0.31 0.31|0.36 0.35|0.52 0.53 0.54 0.53 0.57 1.00
```
Figure 3: Similarity matrix and division of three groups. The yellow part shows "Chlopi" volumes, green - "Lalka" volumes and blue - Sienkiewicz's Trilogy.

The symmetric similarity matrix (see Figure 4) shows individual indices for every pair of books in this scenario. Background sectors are uncoloured and individual foreground groups are uniquely coloured. Heatmap provides a visualisation of Values (see Figure 5) in both background and foreground are stable and their mean representations show the accurate picture. Heatmap also shows different static background intensities for every pair of classes.

Given the results, it is certain that there is a connection between features and lexis, represented by a distinct 20% difference between minimally and maximally feature-related books. Static background noise for mainly unrelated fiction books ranges from 0.28 to 0.39 with mean of 0.33. Foreground values are also static and surprisingly do not range too much in different groups – only by 8%. That implies that differences between book volumes and entries of book series have the same degree.

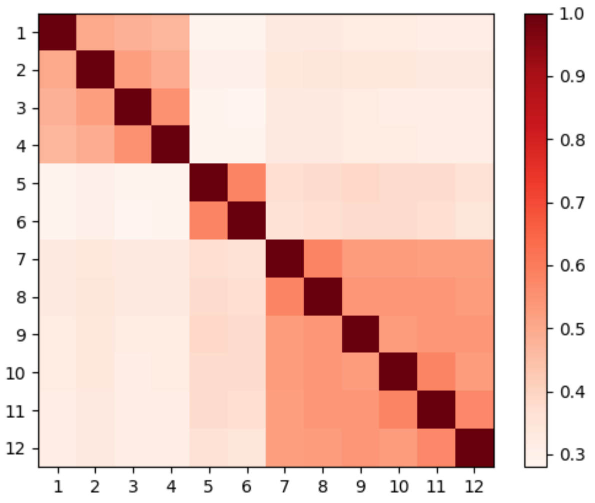


Figure 4: Similarity matrix heatmap.

3.3. Scenario 2 – Author differentiation

The second question is whether this tendency of starker lexical likeness remains between the works of one author compared to the works written by other authors. This scenario targets supposedly the heaviest factor after literature type which influences the book's lexis. To examine this factor three classic Polish authors were selected – Bolesław Prus (7 books), Władysław Reymont (9 books) and Henryk Sienkiewicz (10 books). H. Sienkiewicz's books include all three books of his Trilogy as a reference to absolute levels of similarity. W. Reymont's books also include the "Rok 1794" trilogy, which probably exhibits similar behaviour.

Clustering algorithms mainly recreated supposed division (see Figure 6) but misplaced three of Reymont's books and one Sienkiewicz's book into Prus's cluster. While background noise remains exactly the same (33%) the foreground equals only 40% thus 13% shorter than the practical maximum acquired in the first scenario (53%). It is obvious that the anomaly occurred inside Reymont's cluster, which has only 36,5% compared to 40-42% of two other clusters.

Differences shown in the heatmap (see Figure 6) are not as distinct as in scenario 1 though the background has similar values. An excerpt of high values inside Sienkiewicz's Trilogy (positions 19-21) contrasts with his other books (17-26), which in turn is bordered by smaller background values. Higher values also separate Prus's books (1-7) from the background, but it seems that the three next books (8-10) which belong to Reymont are also part of this cluster. Recreated Prus's cluster holds all these books and also one book written by Reymont (17). All other Reymont's books have an anomaly causing much lower background (less than 30%) and even foreground values around them. The only exceptions are between books from the "Rok 1794" trilogy. Background values between Prus's and Sienkiewicz's clusters are higher (36%) and are harder to separate from their inner similarities (40-42%).

```
Group 1 (r = 0.613; inner similarity - 0.402):
        books_'Dusze_w_niewoli'_-_Prus (43%)
        books_'Anielka'_-_Prus (100%)
        books_'Placowka'_-_Prus (42%)
        books_'Lalka'_-_Prus (46%)
        books_'Emancypantki'_-_Prus (46%)
        books_'Faraon'_-_Prus (40%)
        books_'Dzieci'_-_Prus (42%)
        books_'Komediantka'_-_Reymont (39%)
        books_'Fermenty'_-_Reymont (40%)
        books_'Ziemia_obiecana'_-_Reymont (40%)
        books_'Na_marne'_-_Sienkiewicz (40%)
Group 2 (r = 0.629; inner similarity - 0.421):
        books_'Za_chlebem'_-_Sienkiewicz (37%)
        books_'Ogniem_i_mieczem'_-_Sienkiewicz (56%)
        books_'Pan_Wolodyjowski'_-_Sienkiewicz (100%)
        books_'Potop'_-_Sienkiewicz (58%)
        books_'Bez_dogmatu'_-_Sienkiewicz (37%)
        books_'Rodzina_Polanieckich'_-_Sienkiewicz (42%)
        books_'Quo_vadis'_-_Sienkiewicz (45%)
        books_'Krzyzacy'_-_Sienkiewicz (52%)
        books_'W_pustyni_i_w_puszczy'_-_Sienkiewicz (43%)
Group 3 (r = 0.672; inner similarity - 0.365):
        books_'Chlopi'_-_Reymont (34%)
        books_'Wampir'_-_Reymont (35%)
        books_'Ostatni_sejm_Rzeczypospolitej'_-_Reymont (53%)
        books_'Nil_desperandum'_-_Reymont (100%)
        books_'Insurekcya'_-_Reymont (47%)
        books_'Bunt'_-_Reymont (33%)

Background noise similarity - 0.332
Common inner similarity - 0.403
Foreground similarity - 0.071

Group's similarity matrix:
1.000 0.364 0.303
0.364 1.000 0.309
0.303 0.309 1.000

Group's distance matrix:
0.000 0.636 0.697
0.636 0.000 0.691
0.697 0.691 0.000
```

Figure 5: The result of clustering 26 books into 3 classes.

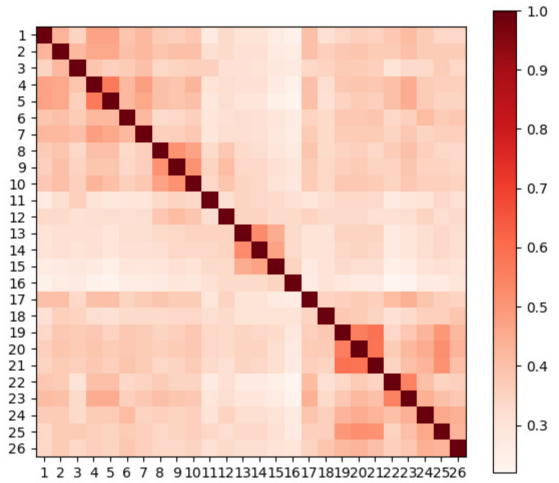


Figure 6: Similarity matrix heatmap. 1 to 7 are Prus's books, 8 to 16 – Reymont's books and 17 to 26 are Sienkiewicz's books.

Overall while Prus's and Reymont's clusters have predictable values, which are at least 4% higher inside them, Reymont's works display a much stronger distance between themselves and to every other book, excluding his early works (8: "Komediantka", 9: "Fermenty", 10: "Ziemia obiecana") which still obey aforementioned tendencies. It may be caused by Reymont's ability to change his style and lexis selection according to his needs resulting in nearly background-level likeness between his books or just by the use of enormous vocabulary replacing a lot of more commonplace synonyms which causes anomalously low background levels. Anyway W. Reymont is known for his thematically, literary and valuably diverse works.

3.4. Scenario 3 – Types of literature differentiation

The third research question reviews one of the most popular beliefs about language reading and comprehension difficulty. Non-fiction literature, especially academic one is believed to be much more demanding to its reader due to a great amount of subject-specific terminology and more sophisticated academic or professional writing.

```
Group 1 (r = 0.790; inner similarity - 0.223):
        books_law_'Kodeks_wyborczy' (22%)
        books_law_'Kodeks_morski' (100%)
        books_law_'Kodeks_pracy' (27%)
        books_law_'Kodeks_wykroczen' (21%)
        books_law_'Kodeks_karny' (21%)
        books_law_'Kodeks_cywilny' (31%)
Group 2 (r = 0.693; inner similarity - 0.327):
        books_'Ziemia_obiecana'_-_Reymont (100%)
        books_'Emancypantki'_-_Prus (40%)
        books_'Quo_vadis'_-_Sienkiewicz (36%)
        books_'Popioly'_-_Zeromski (34%)
        books_'Komornicy'_-_Orkan (31%)
        books_'Prochno'_-_Berent (36%)
Group 3 (r = 0.812; inner similarity - 0.184):
        books_textbooks_'Metody_numeryczne_w_przykladach' (100%)
        books_textbooks_'Fizyka' (19%)
        books_textbooks_'Podstawy_programowania' (19%)
        books_textbooks_'Zastosowanie_programu_Mathcad' (21%)

Background noise similarity - 0.109
Common inner similarity - 0.260
Foreground similarity - 0.151

Group's similarity matrix:
1.000 0.110 0.110
0.110 1.000 0.105
0.110 0.105 1.000

Group's distance matrix:
0.000 0.890 0.890
0.890 0.000 0.895
0.890 0.895 0.000
```

Figure 7: The result of clustering fiction and two types of non-fiction literature into three separate classes.

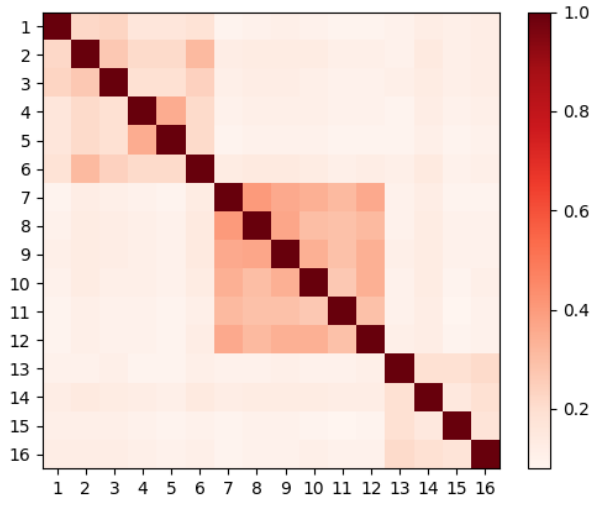


Figure 8: Similarity matrix heatmap. 1 to 6 are codes of law, 7 to 12 – fiction books and 13 to 16 are textbooks.

This scenario compares three sets of books – namely six works of legal literature, six fiction books and four textbooks. All of these books have diversified features, such as different authors or different fields for textbooks.

As shown in Figure 7 clustering had successfully recreated the supposed three groups with 15% of foreground similarity. Inner similarities are falling from 30% inside fiction books through 22% between legal documents down to 18% for textbooks but stay significantly higher than static 11% background noise. These conclusions can be drawn also from heatmap visualisation in Figure 8.

4. Conclusions

The research concluded in this work indicates that lexical similarity can be used to state "feature" distance, starting from the type of literature, author and ending with genres and styles. Thus the compact hypothesis for books' classes and features is proved.

The first scenario analyses how the greatest and the smallest achievable values for the similarity between two arbitrary fiction books correlate with nonexistent and absolute feature similarity. Background for nonexistent similarity is 33% when inner similarity goes above 50%.

The second scenario that evaluates books written by different authors shows that in most cases their works can be recognised by more than 5% of foreground similarity. This relation works unless these authors specifically use much more distinct lexis for every other work as W. Reymont did.

The third scenario shows a successful attempt to separate fiction books, textbooks and legal literature apart. The practically featureless background shows an astonishing 11% similarity which breaks the practical minimum achieved in the first scenario by 22%. It also displays significant differences in similarity coefficients inside these three classes.

Fiction books appear to have a much bigger tendency to share common lexis. Also, it is possible to differentiate types of further divisions inside this group.

Practical absolute similarity values found in this research are 11% for absent feature similarity (including the type of literature) and 53% for an identical set of features. Thus there is a 42% of similarity coefficient range available for research.

The solution presented in this article allows to do basic analysis concerning most outstanding features, but many steps taken in this article can be refined for better results including a more precise dictionary with a better lemma selection procedure and more appropriate similarity coefficient. This is not limited to the Polish language and requires only a change of dictionary complete with declensions and respective lemmas to analyse texts in other languages. Lexis-feature relation appears to be a promising field for further thorough research.

References

[1] R. Singh, S. Singh, Text Similarity Measures in News Articles by Vector Space Model Using NLP, Journal of The Institution of Engineers (India): Series B 102 (2021) 329–338.

[2] A. Huang, Similarity Measures for Text Document Clustering, Proceedings of the Sixth New Zealand Computer Science Research Student Conference 4 (2008) 49–56.

[3] M. B. Magara, S. O. Ojo, T. Zuva, A Comparative Analysis of Text Similarity Measures and Algorithms in Research Paper Recommender Systems, 2018 Conference on Information Communications Technology and Society (2018) 1–5.

[4] A. W. Qurashi, V. Holmes, A. P. Johnson, Document Processing: Methods for Semantic Text Similarity Analysis, In 2020 International Conference on INnovations in Intelligent SysTems and Applications (2020) 1–6.

[5] W. H. Gomaa, A. A. Fahmy, A Survey of Text Similarity Approaches, International Journal of Computer Applications 68 (2013) 13–18.

[6] S. Bekmirzaev, T. H. Kim, B. C. Lee, Pairwise Similarity Analysis and Quality Estimation on Classical Chinese Poetry of Ancient Korea in 15th Century, International Journal of Applied Engineering Research 12 (2017) 13884–13890.

[7] D. M. Kaplan, D. M. Blei, A Computational Approach to Style in American Poetry, In Seventh IEEE International Conference on Data Mining (2007) 553–558.

[8] C. D. Manning, H. Schütze, Foundations of Statistical Natural Language Processing, MIT press, 1999.

[9] R. Grishman, Computational Linguistics: An Introduction, Cambridge University Press, 1986.

[10] R. Grzegorczykowa, R. Laskowski, H. Wróbel, Gramatyka współczesnego języka polskiego. Morfologia, Wydawnictwo Naukowe PWN, 1999.

[11] S. Niwattanakul, J. Singthongchai, E. Naenudorn, S. Wanapu, Using of Jaccard Coefficient for Keywords Similarity, In Proceedings of the International Multiconference of Engineers and Computer Scientists 1 (2013) 380–384.

[12] L. Kaufman, P. J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, John Wiley & Sons, 2009.

[13] Słownik języka polskiego, https://sjp.pl, [18.09.2021].