

Zgłębianie danych - raport

Kacper Siora

15 maja 2014

1 Treść zadania

"Sprawdzić prawo Zipf'a w języku polskim: dla różnych grup dokumentów sporządzić listę występujących słów, uporządkować je wg częstości wystąpień (od najczęstszego) i sprawdzić wartości iloczynów $r \cdot f$ gdzie r jest rangą słowa (numerem na liście), f częstością wystąpień tego słowa. Dla języka angielskiego iloczyn ten wynosi ok. 0.1."

Prawo Zipf'a mówi, że w korpusie języka naturalnego, częstotliwość występowania słów jest odwrotnie proporcjonalna do pozycji w rankingu. Ranking powstaje w wyniku zliczenia częstotliwości występowania słów oraz posortowania malejąco powstałej listy.

Prawo to jest wszechobecne w świecie i odeszło od sensu stricte lingwistycznego. Prawo Zipf'a może być pomocne przy ustalaniu tzw. *stoplist* (*ang. stopwords list*). Ponadto przydatne jest w takich sytuacjach jak: liczba ludności w miastach, częstość nazwisk w książce telefonicznej, rozkład trzęsień Ziemi.

2 Narzędzia

Aby zliczyć ilość słów oraz częstość ich występowania postanowiłem skorzystać z systemu zarządzania bazą danych MongoDB. Korzystałem również ze skryptu do przerabiania tekstu, oraz programu tr w systemie Linux.

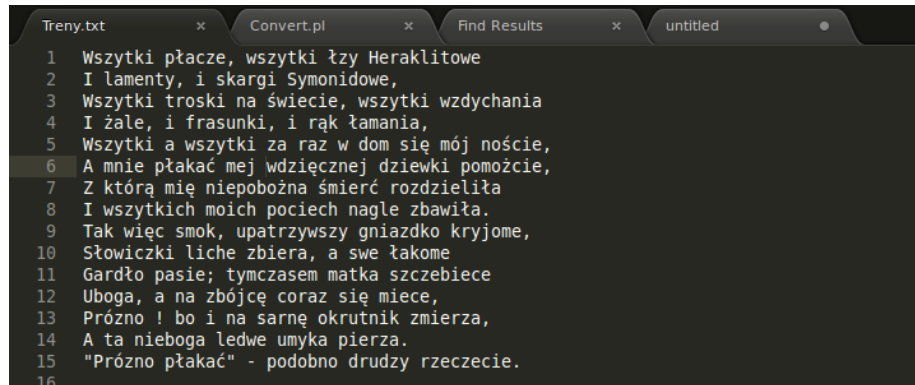
3 Dane

Dokumenty, których używałem w projekcie to:

- *Alicja w Krainie Czarów*
- *Nad Niemnem*
- *W pustyni i w puszczy*
- *Pianista*
- *Ewangelia wg św. Mateusza*
- *Regulamin studiów Uniwersytetu Gdańskiego*
- *Zbiór tekstów z płyty „Marek Grechuta & Anawa”*
- *Treny J. Kochanowskiego*

3.1 Przygotowanie danych

Po ściągnięciu wszystkich ww. dokumentów, musiałem odpowiednio je odpowiednio przerobić.

A screenshot of a text editor window with four tabs: 'Treny.txt', 'Convert.pl', 'Find Results', and 'untitled'. The 'Treny.txt' tab is active, displaying a Polish poem by Jan Kochanowski. The text is as follows:

```
1 Wszystkie płacze, wszystkie łzy Heraklitowe
2 I lamenty, i skargi Symonidowe,
3 Wszystkie troski na świecie, wszystkie wdychania
4 I żale, i frasunki, i rąk łamanie,
5 Wszystkie a wszystkie za raz w dom się mój noście,
6 A mnie płakać mej wdzięcznej dziewczki pomożcie,
7 Z którą mię niepobożna śmierć rozdzieliła
8 I wszystkich moich pociech nagle zbawiła.
9 Tak więc smok, upatrzysz gniazdko kryjome,
10 Słowiczki liche zbiera, a swe łakome
11 Gardło pasie; tymczasem matka szczebiece
12 Uboga, a na zbójcę coraz się miece,
13 Próżno ! bo i na sarnę okrutnik zmierza,
14 A ta nieboga ledwie umyka pierza.
15 "Próżno płakać" - podobno drudzy rzeczecie.
16
```

Przykładowy dokument przed użyciem skryptu

Przy pomocy poniższego skryptu usunąłem wszelkie niepotrzebne znaki oraz pozbyłem się cyfr i polskich liter.

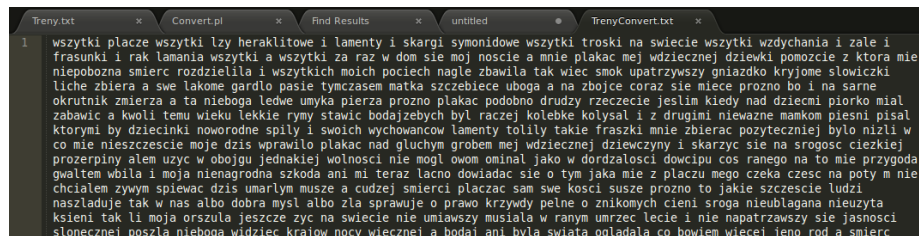
```
#!/usr/bin/perl

while (<>) {
    s/<.*>//;          # usuwanie < >
    s/[ ]//g;          # usuwanie [ ]
    s/[ ]//g;
    $_="$_ " ;
    tr/A-Z/a-z/;        # zamiana na małe litery
    s/([0-9]*)//g;      # usuwanie liczb
    s/ą/a/g;            # zamiana polskich znaków
    s/Ą/a/g;
    s/ć/c/g;
    s/Ć/c/g;
    s/ę/e/g;
    s/Ę/e/g;
    s/ł/l/g;
    s/Ł/l/g;
    s/ń/n/g;
    s/Ń/n/g;
    s/ó/o/g;
    s/Ó/o/g;
    s/ś/s/g;
    s/Ś/s/g;
    s/ż/z/g;
    s/Ż/z/g;
    s/ż/z/g;
    s/Ž/z/g;
    tr/a-z/ /cs;
    chop;
    print $_;
}
```

```
student@virtualbox:~/Pulpit/ksiazki$ perl Convert.pl Treny.txt > TrenyConvert.txt
student@virtualbox:~/Pulpit/ksiazki$
```

Komenda wykonująca skrypt

Po wykonaniu skryptu, z dokumentu usunięte zostały wszystkie znaki specjalne, cyfry oraz znaki interpunkcyjne. Dokument został zapisany słowo po słowie oddzielone spacjami w jednej linii.



Przykładowy dokument po przerobieniu skryptem

Przed importem do bazy Mongo należy wykonać jeszcze jedną komendę. Dzięki programowi *tr* każde słowo w naszym dokumencie zostanie zapisane w oddzielnej linii.

```
student@virtualbox:~/Pulpit/ksiazki$ tr --squeeze-repeats '[:blank:]' '\n' < TrenyConvert.txt > TrenyMongo.txt
student@virtualbox:~/Pulpit/ksiazki$
```

Użycie programu tr do kolejnej konwersji dokumentu

Przykładowy dokument po konwersji programem tr

3.2 Import danych

Tak przygotowane dane, możemy już importować do bazy danych Mongo. Aby zacząć pracę na systemie Windows wystarczy po prostu ściągnąć i rozpakować program ze strony producenta. Na sam początek musimy uruchomić program *mongod.exe*

```
Windows PowerShell
PS C:\mongodb-win32-x86_64-2008plus-2.6.1\bin> .\mongod.exe
C:\mongodb-win32-x86_64-2008plus-2.6.1\bin> mongod.exe --help for help and startup options
2014-05-14T19:58:59.951+0200 [initandlisten] MongoDB starting : pid=7104 port=27017 dbpath=\data\db\ 64-bit host=Sior8ki
2014-05-14T19:58:59.952+0200 [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2014-05-14T19:58:59.953+0200 [initandlisten] db version v2.6.1
2014-05-14T19:58:59.954+0200 [initandlisten] git version: 4b95b086d2374bdcfcd2249272fb552c9c726e8
2014-05-14T19:58:59.955+0200 [initandlisten] build info: windows sys.getwindowsversion(major=6, minor=1, build=7601, pla
tform=2, service_pack='Service Pack 1') BOOST_LIB_VERSION=1_49
2014-05-14T19:58:59.955+0200 [initandlisten] allocator: system
2014-05-14T19:58:59.955+0200 [initandlisten] options: {}
2014-05-14T19:59:00.042+0200 [initandlisten] journal dir=\data\db\journal
2014-05-14T19:59:00.044+0200 [initandlisten] recover : no journal files present, no recovery needed
2014-05-14T19:59:01.892+0200 [initandlisten] waiting for connections on port 27017
2014-05-14T19:59:13.205+0200 [initandlisten] connection accepted from 127.0.0.1:50321 #1 (1 connection now open)
```

Uruchamianie usługi MongoDB daemon

Teraz dopiero możemy zacząć import danych do bazy

```
Windows PowerShell
PS C:\mongodb-win32-x86_64-2008plus-2.6.1\bin> ./mongoimport -d books -c Trency --fields word --type csv --file TrencyMong.txt
connected to: 127.0.0.1
2014-05-14T20:12:55.172+0200 check 9 3776
2014-05-14T20:12:55.172+0200 imported 3776 objects
PS C:\mongodb-win32-x86_64-2008plus-2.6.1\bin>
```

Import danych do bazy

Po zaimportowaniu wszystkich dokumentów możemy sprawdzić ile słów jest w danym tekście

```
> db.Trency.count()
3776
> db.Trency.distinct("word").length
1809
>
```

Komenda MongoDB zwracająca ilość słów

4 Wyniki

Aby obliczyć *f* (częstość wystąpienia) danego słowa, należy podzielić liczbę jego wystąpień przez ilość wszystkich słów w dokumencie.

Dokument	Ilość wszystkich słów	Ilość słów różnych
Alicja w Krainie Czarów	20018	5495
Nad Niemnem	164352	29677
W pustyni i w puszczy	100485	17837
Pianista	49201	13590
Ewangelia wg św. Mateusza	17455	4287
Regulamin studiów	4798	1267
Marek Grechuta & Anawa	1718	768
Treny J. Kochanowskiego	3776	1809

Alicja w krainie czarów				
<i>r (rank)</i>	słowo	ilość wystąpień	<i>f (częstość wystąpienia)</i>	$r * f$
1	<i>się</i>	655	0,0327	0,0327
2	<i>nie</i>	494	0,0247	0,0494
3	<i>i</i>	454	0,0227	0,0680
4	<i>w</i>	409	0,0204	0,0817
5	<i>na</i>	388	0,0194	0,0969
6	<i>alicja</i>	339	0,0169	0,1016
7	<i>ze</i>	322	0,0161	0,1126
8	<i>z</i>	319	0,0159	0,1275
9	<i>to</i>	317	0,0158	0,1425
10	<i>do</i>	225	0,0112	0,1124
11	<i>po</i>	144	0,0072	0,0791
12	<i>tak</i>	131	0,0065	0,0785
13	<i>a</i>	126	0,0063	0,0818
14	<i>o</i>	121	0,0060	0,0846
15	<i>co</i>	116	0,0058	0,0869
Średnia $r*f$ =				0,0891

Nad Niemnem				
<i>r (rank)</i>	słowo	ilość wystąpień	<i>f (częstość wystąpienia)</i>	$r * f$
1	<i>i</i>	7475	0,0455	0,0455
2	<i>się</i>	3974	0,0242	0,0484
3	<i>z</i>	3545	0,0216	0,0647
4	<i>w</i>	3533	0,0215	0,0860
5	<i>na</i>	2885	0,0176	0,0878
6	<i>nie</i>	2744	0,0167	0,1002
7	<i>ze</i>	1744	0,0106	0,0743
8	<i>a</i>	1577	0,0096	0,0768
9	<i>do</i>	1576	0,0096	0,0863
10	<i>to</i>	1365	0,0083	0,0831
11	<i>ale</i>	1051	0,0064	0,0703
12	<i>jej</i>	1034	0,0063	0,0755
13	<i>o</i>	979	0,0060	0,0774
14	<i>jak</i>	954	0,0058	0,0813
15	<i>po</i>	783	0,0048	0,0715
Średnia $r*f$ =				0,0753

W pustyni i w puszczy				
<i>r (rank)</i>	słowo	<i>ilość wystąpień</i>	<i>f (częstość wystąpienia)</i>	r * f
1	<i>i</i>	4050	0,0273	0,0273
2	<i>sie</i>	2747	0,0273	0,0547
3	<i>w</i>	2241	0,0223	0,0669
4	<i>na</i>	1993	0,0198	0,0793
5	<i>nie</i>	1958	0,0195	0,0974
6	<i>z</i>	1761	0,0175	0,1052
7	<i>ze</i>	1597	0,0159	0,1113
8	<i>do</i>	1208	0,0120	0,0962
9	<i>a</i>	1046	0,0104	0,0937
10	<i>to</i>	854	0,0085	0,0850
11	<i>ale</i>	785	0,0078	0,0859
12	<i>po</i>	731	0,0073	0,0873
13	<i>stas</i>	630	0,0063	0,0815
14	<i>nel</i>	549	0,0055	0,0765
15	<i>jak</i>	536	0,0053	0,0800
Średnia r*f =				0,0819

Pianista				
<i>r (rank)</i>	słowo	<i>ilość wystąpień</i>	<i>f (częstość wystąpienia)</i>	r * f
1	<i>sie</i>	1530	0,0311	0,0311
2	<i>w</i>	1441	0,0293	0,0586
3	<i>i</i>	1398	0,0284	0,0852
4	<i>z</i>	1074	0,0218	0,0873
5	<i>na</i>	1036	0,0211	0,1053
6	<i>nie</i>	877	0,0178	0,1069
7	<i>do</i>	665	0,0135	0,0946
8	<i>ze</i>	542	0,0110	0,0881
9	<i>to</i>	327	0,0066	0,0598
10	<i>po</i>	303	0,0062	0,0616
11	<i>a</i>	271	0,0055	0,0606
12	<i>o</i>	265	0,0054	0,0646
13	<i>jak</i>	253	0,0051	0,0668
14	<i>gdy</i>	250	0,0051	0,0711
15	<i>już</i>	235	0,0048	0,0716
Średnia r*f =				0,0742

Ewangelia wg św. Mateusza				
<i>r (rank)</i>	słowo	<i>ilość wystąpień</i>	<i>f (częstość wystąpienia)</i>	r * f
1	<i>i</i>	800	0,0458	0,0458
2	<i>nie</i>	382	0,0219	0,0438
3	<i>sie</i>	356	0,0204	0,0612
4	<i>do</i>	328	0,0188	0,0752
5	<i>a</i>	306	0,0175	0,0877
6	<i>w</i>	294	0,0168	0,1011
7	<i>na</i>	266	0,0152	0,1067
8	<i>ze</i>	241	0,0138	0,1105
9	<i>jest</i>	167	0,0096	0,0861
10	<i>to</i>	151	0,0087	0,0865
11	<i>go</i>	126	0,0072	0,0794
12	<i>jesus</i>	123	0,0070	0,0846
13	<i>gdy</i>	120	0,0069	0,0894
14	<i>lecz</i>	115	0,0066	0,0922
15	<i>mu</i>	109	0,0062	0,0937
Średnia r*f =				0,0829

Regulamin studiów				
<i>r (rank)</i>	słowo	<i>ilość wystąpień</i>	<i>f (częstość wystąpienia)</i>	r * f
1	<i>w</i>	245	0,0511	0,0511
2	<i>i</i>	100	0,0208	0,0417
3	<i>studiów</i>	99	0,0206	0,0619
4	<i>z</i>	96	0,0200	0,0800
5	<i>na</i>	90	0,0188	0,0938
6	<i>moze</i>	68	0,0142	0,0850
7	<i>do</i>	68	0,0142	0,0992
8	<i>lub</i>	54	0,0113	0,0900
9	<i>dziekan</i>	51	0,0106	0,0957
10	<i>student</i>	47	0,0098	0,0980
11	<i>sie</i>	45	0,0094	0,1032
12	<i>przez</i>	43	0,0090	0,1075
13	<i>studenta</i>	43	0,0090	0,1165
14	<i>ust</i>	41	0,0085	0,1196
15	<i>egzaminu</i>	38	0,0079	0,1188
Średnia r*f =				0,0908

Marek Grechuta & Anawa				
<i>r (rank)</i>	słowo	<i>ilość wystąpień</i>	<i>f (częstość wystąpienia)</i>	r * f
1	w	58	0,0338	0,0338
2	nie	56	0,0326	0,0652
3	i	41	0,0239	0,0716
4	sie	36	0,0210	0,0838
5	to	33	0,0192	0,0960
6	kto	32	0,0186	0,1118
7	na	24	0,0140	0,0978
8	jak	23	0,0134	0,1071
9	z	22	0,0128	0,1153
10	ty	18	0,0105	0,1048
11	ze	15	0,0087	0,0960
12	czy	15	0,0087	0,1048
13	a	14	0,0081	0,1059
14	jest	13	0,0076	0,1059
15	pierwszy	12	0,0070	0,1048
Średnia r*f =				0,0936

Treny				
<i>r (rank)</i>	słowo	<i>ilość wystąpień</i>	<i>f (częstość wystąpienia)</i>	r * f
1	nie	117	0,0310	0,0310
2	i	90	0,0238	0,0477
3	w	82	0,0217	0,0651
4	a	80	0,0212	0,0847
5	sie	74	0,0196	0,0980
6	na	50	0,0132	0,0794
7	z	41	0,0109	0,0760
8	co	39	0,0103	0,0826
9	tak	34	0,0090	0,0810
10	ze	24	0,0064	0,0636
11	to	24	0,0064	0,0699
12	ani	21	0,0056	0,0667
13	ty	21	0,0056	0,0723
14	jako	20	0,0053	0,0742
15	mi	19	0,0050	0,0755
Średnia r*f =				0,0712


```
Windows PowerShell
PS C:\mongodb-win32-x86_64-2008plus-2.6.1\bin> .\mongo.exe
MongoDB shell version: 2.6.1
connecting to: test
> use books
switched to db books
> show collections
Alicja
EwangeliaMat
Grechuta
NadNiemnem
Pianista
Pustynia
Reg
Treny
system.indexes
> db.Treny.aggregate({$group: {_id: "$word", count: {$sum: 1}}}, {$sort: {count:-1}}, {$limit: 15})
{ "_id" : "nie", "count" : 117 }
{ "_id" : "i", "count" : 90 }
{ "_id" : "w", "count" : 82 }
{ "_id" : "a", "count" : 80 }
{ "_id" : "sie", "count" : 74 }
{ "_id" : "na", "count" : 50 }
{ "_id" : "z", "count" : 41 }
{ "_id" : "co", "count" : 39 }
{ "_id" : "tak", "count" : 34 }
{ "_id" : "ze", "count" : 24 }
{ "_id" : "to", "count" : 24 }
{ "_id" : "ani", "count" : 21 }
{ "_id" : "ty", "count" : 21 }
{ "_id" : "jako", "count" : 20 }
{ "_id" : "mi", "count" : 19 }
>
```

Wyszukiwanie najczęściej występujących słów

5 Wnioski

Najczęściej występujące słowa to oczywiście zaimki, przyimki itp. Są to słowa, które nie niosą ze sobą żadnych przydatnych informacji, dlatego właśnie prawo Zipf'a może być pomocne przy tworzeniu stoplist. Wysoko są również imiona głównych postaci z tekstów (Alicja z *Alicji w Krainie Czarów* czy też Staś i Nel z *Pustyni i w puszczy*). Takich wyników należało się spodziewać.

Iloczyn r^*f spada wraz ze wzrostem słów w tekście, dla testowanych małych dokumentów osiąga on nieco ponad 0,09. Tymczasem dla większych dokumentów spada nawet do 0,07. Średnia ze średnich r^*f wszystkich dokumentów równa się **0,0825**. Można więc się spodziewać, że podobnie jak dla języka angielskiego, tak jak i dla polskiego generalnie iloczyn ten jest blisko 0,1.

6 Bibliografia

Bibliografia:

1. mgr inż. Przemysław Sołdacki, Zastosowanie metod płaskiej analizy tekstu do przetwarzania dokumentów w języku polskim, Warszawa 2006
2. <http://pl.wikipedia.org/>
3. http://www.ccs.neu.edu/home/ekanou/ISU535.09X2/Handouts/Review_Material/zipfslaw.pdf
4. http://panoramix.ift.uni.wroc.pl/pluginfile.php/217/mod_folder/content/1/stare/fizkomputerowa/wyk%EF%BF%BDad5.pdf

Narzędzia:

1. <http://wbzyl.inf.ug.edu.pl/nosql/> - przydatne polecenia
2. <http://www.mongodb.org/> - system zarządzania bazą danych
3. <http://mattmahoney.net/dc/textdata.html#appendixa> – skrypt do „oczyszczania” tekstu

Dane:

1. <http://chomikuj.pl/> - utwory literackie
2. <http://teksty.org/> - teksty piosenek
3. <http://adonai.pl/download/dokumenty/> - Ewangelia
4. <http://mfi.ug.edu.pl/> - regulamin studiów