Paper 1652-2014

Text Mining Reveals the Secret of Success: Identification of Sales Determinants Hidden in Customers' Opinions

Rafał Wojdan, Warsaw School of Economics

ABSTRACT

Nowadays, in the Big Data era, Business Intelligence Departments collect, store, process, calculate, and monitor massive amounts of data. Nevertheless, sometimes hundreds of metrics built on the structured data are inefficient to explain why the offered deal sold better or worse than expected. The answer might be found in text data that every company owns and yet is not aware of its possible usage or neglects its value. This project shows text mining methods, implemented in SAS® Text Miner 12.1, that enable the determination of a deal's success or failure factors based on in-house or Internet-scattered customers' views and opinions. The study is conducted on data gathered from Groupon Sp. z o.o. (Polish business unit) - e-commerce company, as it is assumed that the market is by and large a customer-driven environment.

INTRODUCTION

The demand for information and knowledge increases every year and Business Intelligence is responsible for satisfying it. On paper BI defined as process that makes intelligent use of company available data leading to enhanced decision making and thus competitive advantage, seems to be prepared to keep up with changes and fulfill its role well (De Ville, 2001; Bergeron and Hiller 2002). However, the commonly utilized resources and techniques such as data warehouses, multidimensional, reporting and predictive analyses, narrow the scope of Business Intelligence (Berry and Linoff, 2004). The current design of BI is restricted only to structured data and hence even such powerful tool like data mining can only answer these questions: who, what, when, where, and how much (Gao, Chang and Song, 2005). The remaining and most important question why can be answered with text data. On the 'getting data in' side of BI process data warehouse is not designed to store semi-or unstructured data and on 'getting data out' side data mining cannot analyze this type of data (Gao, Chang and Song, 2005). Hence, there is a need for document warehouses (Ishikawa, 1999; Nassis et al., 2004) and text mining tools usage in order to refresh BI methods and go beyond the previously mentioned limitations. The time is perfect for textual analyses, as the methods are young and leave much space for innovations and there is a huge amount of texts, documents and messages, if not owned by company then available in the Internet, to be mined. This research presents where to look for text data, how to extract information and answer the question why, for example sales is lower or higher than expected. It appears that document warehouses is not indispensable in order to expand Business Intelligence arsenal of analysis and reports with text mining.

The paper shows how BI can benefit from expanding its scope of analysis to text mining. The data used in the study has been made available by Groupon Sp. z o.o. - one of the leading companies in e-commerce business. It is argued here that e-commerce market has great potential for text mining utilization and can be used as fantastic example of how to improve BI with text mining. Compared to banking or insurance industries, where customers' are obliged to share their personal details, e-commerce companies do not possess so much information about their clients as they need to balance between their needs for information and customers' willingness to reveal the information. Yet still paradoxically in banking or insurance sector the latest analysis tools as text mining are almost immediately implemented. It is commonly known that people are not very prone to fill in registration forms. Nonetheless, people are less restrained to express or share their opinions, especially when they feel anonymous or try to achieve a given purpose. Therefore complaint emails, inquiry emails, surveys or product reviews are the best source of information for e-commerce about their clients. Moreover, as e-commerce business attracts clients directly via Internet; its target audience is composed of Internet savvy users, who on a daily basis use forums or social networks sites for expressing their opinions. Keeping this in mind, e-commerce can be considered as one of the most customer opinion driven market. Thus, for companies operating in this market it is crucial to get to know customers' needs and adjust the offer to meet their demands as fast as possible.

In this study it is shown how to use customer care email data, survey data and the Internet scattered data in order to find potential determinants of sales success and failures. In other words how text mining can be used to refine quality of deals and improve future sales. As Groupon is a sales platform, for mid- and small businesses, offering deals at discount, the work presents how to exploit text mining in real business case. Unfortunately, the sales model and other sales metrics are not available for this study, hence obtained variable are potential determinants and require further verification. The work is inspired by Data Science and Business Intelligence approaches, thus it presents whole flow of data from source through processing until final analysis.

DATA

As already mentioned the data used for text mining analysis comes from three sources: customers' surveys, customers' emails and the Internet.

Survey data are gathered from survey filled by the customers who did a purchase on Groupon and stored in a database. Beside multiple choice questions the survey contains typical open question asking for recommendation, opinion or feedback. These written answers are perfect materials for the analysis

Customer Care emails comprise of both complaint and inquiry emails. The data are collected, stored and monitored using Zendesk software. Compared to survey data exporting email contents is more complex and requires use of Zendesk API.

The Internet data are surprisingly strongly scattered. There are many posts about Groupon purchases on various forums, however most frequently there is just one topic in the whole forum referring to Groupon product. There is only one site dedicated for posting comments on Groupon purchases called *antygroupon* and as the name suggests it is focused on collecting only descriptions of negative experiences. Moreover, there are many comments referring to other e-commerce platforms or comments that just criticize other users of this site. Still the data extracted from the site may be useful for identifying negative sentiment and to analyze how such anti-site may affect sale. As the heavy dispersion of text data exists, building a tool that monitors Internet sites is necessary and dedicated crawlers cannot be used. Hence, the only *antygroupon* site is used as an example.

Sales data – gross booking is used as a metric of sales performance. The data are partitioned by product categories so it can be checked if relation between sales and opinions differ between product groups. The sales results are divided by benchmark in order to provide percentage realization of threshold. Other sales metrics or sales model are not available for this study so the main focus is to verify potential correlation between sales and different comments.

DATA EXTRACTION

Every source of data requires individual extraction approach.

Zendesk provides REST API to retrieve data and therefore the SAS® PROC HTTP procedure is used to get the data. In one communication with the API a user can download 100 tickets, so the procedure is used in the macro loop. The collected data are in JSON format and needs to be converted to format that can be easily transformed to datasets. The latest version SAS® Base 9.4 offers PROC JSON procedure that can be used to this, however the study is based on SAS® Base 9.3 and the proposed solution exploits PROC GROOVY procedure, following the answer to the problem posted by Falko Schulz posted on SAS blogs (Schulz, 2013). PROC GROOVY contains submit blocks same as macro structure and due to this PROC GROOVY can be implemented to macro only via %INCLUDE statement. This way JSON data are converted to csv file which are next appended to previously prepared table. The final version of macro contains also commands that clear log and groovy memory so that the macro does not stop due to lack of memory resources. The macro and PROC GROOVY codes can be found in the Appendix 1.

In order to import **Internet data** FILENAME URL, along with DATA STEP containing INFILE statement and INPUT function is used. The data are in HMTL format which happens to require cleaning before further transformation. So the first step is to clean HTML, next convert to XML and finally map to columns, thus achieving final table. Almost all this is done with just one PROC GROOVY procedure that allows to get csv file, which is then imported using simple PROC IMPORT procedure. The FILENAME, DATA STEP and PROC GROOVY are in the appendix 2.

Due to the fact **survey data** are already in csv file, PROC IMPORT statement is enough to get the dataset.

DATA PARSING AND CLEANING

The obtained datasets from Internet or email data still demands parsing and cleaning. PRX prefix regular expressions SAS functions are really efficient in doing this. For example, PRXMATCH can be used to separate email message from everything that provides no value for text analysis like footer or any other text that comes usually after honorifics like 'Best regards' or 'Sincerely yours'. As people sometimes forget to use capital letter the 'i' option is used to search not case-sensitive terms. The below code give hints how to do this:

```
data cleaned;
format parsed $varying7000.;
format trash $varying7000.;
set text;
l=length(trim(email));
p1=prxmatch('/(Best regards)|(Sincerely)/i', email)
if p1>1 then do;
```

```
parsed=trim(substr(description,1,p1-1));
trash=trim(substr(description,p1,l-p1+1));
run;
```

There are also cases when multiple parsing is necessary as it is for customer care date where email history for one ticket is merged in one text. The repeated term is ‘wrote by:’ and it can be used to parse with PRXNEXT function as in the example:

```
DATA parsed;
set email;
l=length(description);
IF _N_ = 1 THEN nap = PRXPARSE('/(napisał)|(pisze:)/i');
RETAIN nap;
START = 1;
STOP = LENGTH(email);
CALL PRXNEXT(NAP,START,STOP,email,POSITION,LENGTH);
ARRAY PR[5];
ARRAY LR[5];
DO I = 1 TO 5 WHILE (POSITION GT 0);
PR[I]=position;
LR[I]=length;
CALL PRXNEXT(Nap,START,STOP,email,POSITION,LENGTH);
END;
if PR1^=. then
parsed=trim(substr(email,1,PR[1]-1));
if PR2^=. and PR2-PR1>10 then
parsed2=trim(substr(email,PR[1]+LR[1],PR[2]-PR[1]-LR[2]-1));
  ⋮

if PR5^=. then
parsed5=trim(substr(comment,PR[4]+LR[4],PR[5]-PR[4]-LR[5]-1));
rest=trim(substr(comment,PR[I-1]+LR[I-1],l-PR[I-1]-LR[I-1]+1));
keep email parsed parsed2 parsed3 parsed4 parsed5 rest;
run;
```

The PRXNEXT works like as PRXMATCH but in a loop, hence it needs start and stop conditions. In every loop it finds an expression and puts its position and length. Then searches remaining text. The last IF conditions used in the above instance returns fifth parsed text but also the rest of text after last ‘wrote by:’ is found by PRXNEXT function.

Additionally, the text can contain unnecessary characters. In case of Internet data despite cleaning HTML the mapping is not perfect and within text date of posting a comment or ‘Reply to comment’ text remains. If the text is constant then TRANWRD function can be used but if text changes as dates of posting then PRXCHANGE should be used. Worth to be noted is the fact, that format of the new variable needs to be specified because during cleaning format is lost. Below is the example:

```
data cleaned;
format text_fin $varying4556.;
set text;
text1=prxchange('s/\d{1,2}\s*(2012)|\d{1,2}\s*(2011)|\d{1,2}\s*(2013)|\d{1,2}
\s*(2014)|//',-1,text);
text_fin=tranwrd(text1, 'Reply to comment','');
run;
```

DATA IDENTIFICATION

One of the difficulties of Text Mining in terms of sales analysis is to match particular opinion with its counterpart sales. There is no problem with survey data which are already identified but some methods have to be applied to customer care text data and especially Internet data. In terms of the first one only historically relatively old ticket require identification. It is done by searching for expressions within emails that meet the conditions of product’s id. Combination of CALL PRXPARSE routine and SUBSTR function is used to do this. If for instance id has 5 to 10 digits then the code looks like this:

```
data matched;
patternID=prxparse('/\d{5,10}/');
call prxsubstr(patternID,text,position1,length1);
if position1>0 then
id=substr(text, position1, length1);
keep text id;
run;
```

Matching Internet data with sales appears to be much more complex task. One of the ideas is to calculate cosine similarity between texts. In Text Mining methodology vector space approach (Salton's,1971) is used to represent text as a vector of terms. Every word in the vocabulary of analyzed collection becomes an independent dimension. In the obtained high dimensional space every text can be represented as vectors. In order to measure numeric similarity between the vectors cosine of the angle between two vectors is calculated (Singhal, 2001). The SAS® procedure that calculates cosine similarity is PROC DISTANCE. However, before texts can be measured with it they need to be represented as vectors. The first step to do this is to parse search collection of document using PROC TGPARSE. Next PROC TMUTIL calculates term weights (weight termwgt=entropy) and cell weights (weight cellwgt=log) for parsed terms and excludes terms that appear in less than 4 texts (reducef=4). The table with start list terms is obtained by joining outputs from both procedures (right join where tmutil output is a right table). Afterwards, searchable text collection is appended to search documents and new collections is parsed where start list is taken from start_list table created in previous step. Then weights are computed for new terms using PROC TMUTIL and results are sorted by term's child id (_TERMNUM_). In order to decrease dimensionality of the achieved matrix Singular Value Decomposition method is used. (SVD) (Deerwester et al., 1990; Albright 2004). The method summarizes textual information into SVD dimensions k by grouping together similar terms into distinct concepts k. As a result of this truncated decomposition, highly dimension matrix is projected on orthonormal k-dimensional subspace which is the best fit to represent the data set (SAS Institute Inc., 2012). The SAS® function dedicated for computing SVD is PROC SPSVD. Finally, PROC DISTANCE with method=cosine can be used to compute diagonal quadratic matrix comprised of measures for every pair of documents. The size of matrix can be problematic if there is a lot of documents to compare, as for example approximately 50 000 documents gives table of size around 22 GB. Therefore, it is recommended to use macro that takes the partition of searched documents and appends to it searchable documents. Then it calculates distance and creates a table with only these pairs that meet the conditions, for example metrics between 0.7 and 1. One should be excluded because it is the value that is located on diagonal so only between pair composed of the same document. The whole code along with the macro can be found in the appendix 3. Although the method has mathematical background and is commonly used in modern information retrieval methodologies (Singhal, 2001), the results for this case are quite disappointing, even when only nouns are used for the analysis. The misclassification for finding correct product is 70% and product from correct category 40%. Consequently, the presented method requires further work and improvements.

The second idea is to use Custom Search Google API as shown in paper "Search Engine using SAS" by Pramod.R (in 2012. Custom Search Google allows to create search profile of chosen sites. Next the received keys can be used in PROC HTTP to communicate with API and get the queried results. It can be done with the macro included in Appendix4, however if the searchable text has special characters then they need to be changed to HTML special characters, so instead of spaces there need to be %20 signs. The only remaining issue is that there is only 100 free search queries per day, hence due to time constraints it is no possible to match antygroupon texts with counterpart products with this method. Nevertheless, the results on tested sample seems promising between 10% and 20% of misclassifications.

Due to above mentioned limitations of both tested methods opinions from *antygroupon* Internet site cannot be used in sales analysis.

TEXT MINING ANALYSIS

However, collecting, extracting and structuring data coming from different sources is already a challenging part; the obtained bag of words is still far from being a form of text that can be analyzed. There are couple of steps, common for every text mining study, that comprise pre-processing which is the first part of the section. In the second part text analytics methods are presented.

TEXT FEATURES

The first step is parsing. During this step the raw text is cleaned which means special characters and punctuations are separated from words. Next documents are broken into tokens which are words, symbols, phrases and meaningful elements. At this point, whole text is converted to lower case, so if the analysis requires case sensitiveness then the special entity needs to be defined. After that, part of speech of every term is identified based on its context and every word is summarized as informative or non-informative part of speech. For instance, nouns

and adjectives are informative and conjunctions, auxiliaries or prepositions are non-informative lexical category. Furthermore, the irrelevant parts of speech can be excluded from further processing. Finally, words are simplified to basic form of speech which is called stemming. In other words, all forms of term are eliminated what significantly decreases number of words for further evaluation. The Parse Node of SAS Text Miner 12.1 allows also setting synonyms, stop and start list specifications. The English synonyms base is implemented in Text Miner, whereas for other languages the users need to create their own dictionary. Stop list is useful in eliminating text elements with very low or even zero informative value such as 'a' or 'the', while start list limits the analysis to only chosen words.

The second step is filtering. Filtering allows for further reduction of 'useless' words. The idea is that every word that will not aid text classification should be ignored, therefore terms with within document frequency below specified value are not taken into consideration for further analysis. The filter node supports also spelling check, searching for expressions, sub-setting documents and specifying minimum number of documents and maximum number of terms. However, the most interesting option for non-English analysis is filter viewer which enables interactive terms choice, spell checking and synonyms creation. It supports also analysis of terms' relationship. The outcome of above methods is high-dimensional term-by-document matrix composed of raw frequency of term appearance in a document (Coussement, 2008).

Weighting by importance of term in the whole corpus is needed in order to correct for within document frequency (Sparck Jones, 1973). The measures that reflect importance of term in document and importance of term in all documents collection are respectively term frequency (tf) and inverse term frequency (idf). The equations for the weights are following:

1) $\mathrm{tf}_{ij} = \log_2(\mathrm{n}_{ij} + 1)$
where i - term, j - document, nij – frequency of term i in j

2) $\mathrm{idf}_i = \log_2\left(\frac{n}{\mathrm{tf}_{ij}}\right) + 1$
where i – term, n – frequency of all documents

Thus, the weight of term i in document j (wij) is given by

3) $w_{ij} = \mathrm{tf}_{ij}\mathrm{idf}_i$

The intuition behind these metrics is that weighted importance of i increases proportionally with the frequency of i in j and decreases with the frequency of i in whole set of documents. (Coussement, 2008). Eventually, weighted term-by-document matrix is obtained.

The problem that still exists is high dimensionality of the achieved matrix, as there are many distinct corpus terms. The already mentioned SVD method is used to decrease high dimensionality. The method summarizes textual information into SVD dimensions k by grouping together similar terms into distinct concepts k. As a result of this truncated decomposition, highly dimension matrix is projected on orthonormal k-dimensional subspace which is the best fit to represent the data set (SAS Institute Inc., 2012). The pivotal point in performing the transformation, which affects the properties of obtained subset, is decision upon number of k. As rule of thumb, high number of k gives more precisely projection but require high computing resources. Generally, it is agreed that values of between 2 and 50 are useful for clustering and between 30 and 200 for classification and prediction (Sanders, 2004).

After pre-processing text data can be used for various analyses. Categorization can be performed by using Clustering or Topic Node. The clustering categorizes documents into disjoint clusters based on SVD factors. Two algorithms can be assigned for this task: Expectation Maximization and Hierarchical algorithm. The first one results in flat representation of clusters and the second on in hierarchical structure (SAS Institute Inc., 2012). Another way to do categorization is via topic grouping. In contrary to clustering, topic grouping is based on the score assigned to each term and document to each topic. The strength of association is based on set thresholds, thus unlike in clustering here a term can belong to more than one group (SAS Institute Inc., 2012). Furthermore, topic grouping allows for defining own topics.

SENTIMENT ANALYSIS

The same as categorization, sentiment analysis can also be performed in two ways, through classification model or topic grouping. SAS Text Miner offers AFFIN_SENTIMENT dataset built based on AFINN sentiment publically available English sentiment lexicon (SAS Institute Inc., 2012). The lexicon is composed of words assigned to 'Positive' and 'Negative' topic which can be used as user topics in topic grouping. As the text analyzed in the study is in Polish such lexicon requires to be created manually. However, the main approach in research is to develop the classification model. For this purpose a sample of reviews is defined as 1 'Positive', 0 'Neutral' and 2 'Negative'.

Contrary to sampling methods in traditional statistics, in text mining there are no rules regarding the sample size. Sales analysis. The method used in this work is proportional allocation is used which means that total sample size is allocated among the strata in proportion to strata size. It is performed using PROC SURVEYSELECT where strata is MM-YYYY date and sample size is 5000. The sample is used to train Regression, Decision Tree and Neural Network model. Then based on misclassification criteria model is chosen and the remaining data are scored. Below figure present an example of sentiment analysis diagram:

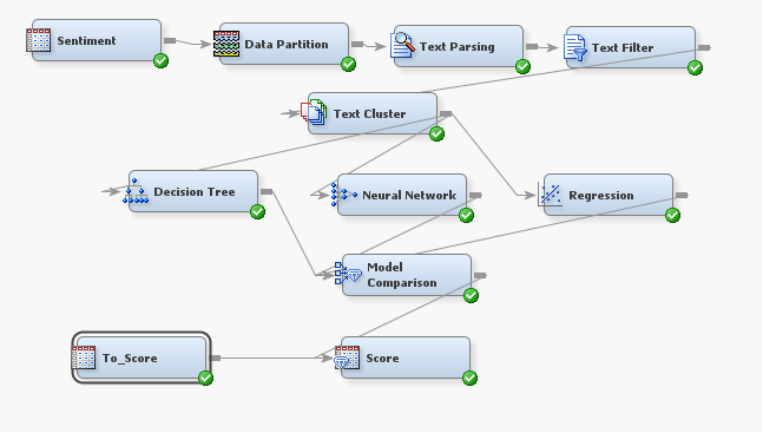


Figure 1. Sentiment analysis diagram

SALES ANALYSIS

The focus of this study is on veryfying if there is any relation between sales and opinions of different sentiment from different sources and if relation exsits then to provide recommendations about possible nature of it and potential usage in further analyses. Therefore, simple regression model without category distinction and models grouped by category are built with SAS® Enterprise Guide 5.1. Additionally, simple decision tree using SAS® Text Miner 12.1 is built in order to provide straightforward decision rules. In both cases explanatory variables are the numbers of opinions from different sources with different sentiment and dependent variable is a percentage realization of the chosen gross booking benchmark.

RESULTS

The performed Text Mining Analysis allows to determine clusters for every data source. The clusters performed better in revealing the characteristic features of texts within data source. As a result, it is easy to distinguish sources one from another. Additionally, the descriptive terms give intuition upon labels that can be assigned to every data source. Below are examples of possible interpretations**:**

	clus_desc	_CLUSTER_	percent
1	+bardzo +miły +pani +smacznie dziękujemy dziękuję jeszcze miłą obsługę pewno pozdrawiam raz serdecznie super sushi	8	0.1033352203
2	+duży +kawa +kuchnia +potraw +smak +wybór +zupa ciasta ciekawe dania dań desery duże menu pierogi	18	0.0860458534
3	+brak +dużo +fitness +możliwość +oferta +profesjonalnie +sposób +szybko ciekawe oferty prowadzone różnorodność są więcej zajęcia	17	0.0772006793
4	+bardzo +dobry +jedzenie +klimat +obsługa +wystrój dobre duże jedzenia miła podane porcje pyszne smaczne świeże	15	0.0574110765

Label 1: Pleasant sushi time – kind service crucial

Label 2: Many courses to choose from, good coffee, innovative dishes – ingredients of nice dining experience

Label 3: Fitness – wide offer of trainings, professional staff, quickly, diversely conducted trainings

Label 4: Very good food, nice ambience, big portions, nicely served dishes – keys to feel you are in an exclusive restaurant

Figure 2. Clusters created by SAS Text Miner® 12.1 and their labels for the Survey text data

	clus_desc	_CLUSTER_	percent
1	+paczka +status utworzona została +ktoś otrzymał zamówiłam przesyłki otrzymałam jeszcze czekam listopada dziś inlogistyki inlogistyka	6	0.1199649737
2	ok wszystko warto było zamówiłam również dzisiaj witam jest grouponu nawet tym buty zabiegi +status	4	0.0954465849
3	+kosmetyczka +zabieg zabiegu zabiegów zabiegi odradzam +salon sobie zdecydowanie byłam trzeba tym polecam nawet zł	9	0.0402802102

Label 1: Problems with shipment

Label 2: Positive experience despite longer waiting time

Label 3: Negative visit at beauty parlor

Figure 3. Clusters created by SAS Text Miner® 12.1 and their labels for the Internet text data

	clus_desc	_CLUSTER_	percent
1	+numer +ul +ważny +zabezpieczający +zrezygnować +zwrot /groupon chciałabym code dniu groupon grouponu hotelu kwoty ma	1	0.1681096228
2	+bardzo +konto +kupon +moje +nr +numer +ul +ważny +zabezpieczający +zrezygnować +zwrot bankowe chciałabym dniu groupon	2	0.1565868998
3	+bardzo +co +pytanie +telefon chciałabym gruponu jak ma mam mi mnie mogę nie niestety państwa	3	0.1465993985
4	+brak +forma +local +nazwa +opis +powód +pracownik +sama +shopping +zamówienie bankowe deal dni grouponu id	4	0.142330455

Label 1: Resignation – cannot make hotel reservation

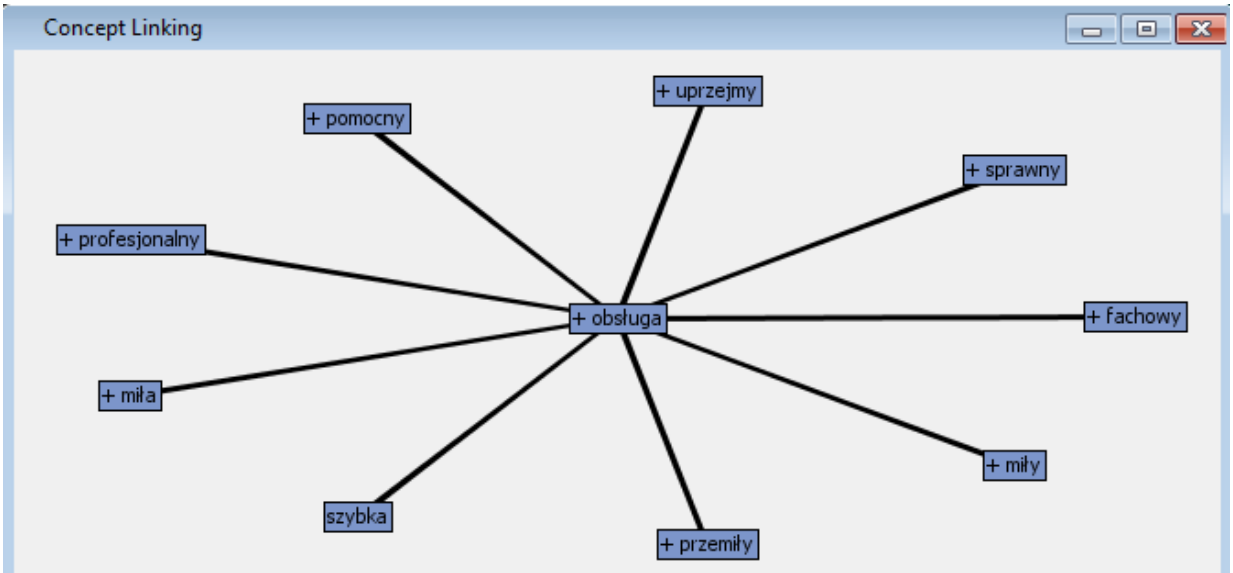
Label 2: Resignation within lawful 10 days – no reason

Label 3: Problem with contacting Customer Department

Label 4: Phone resignation of shopping product – standard form of a ticket prepared by employee

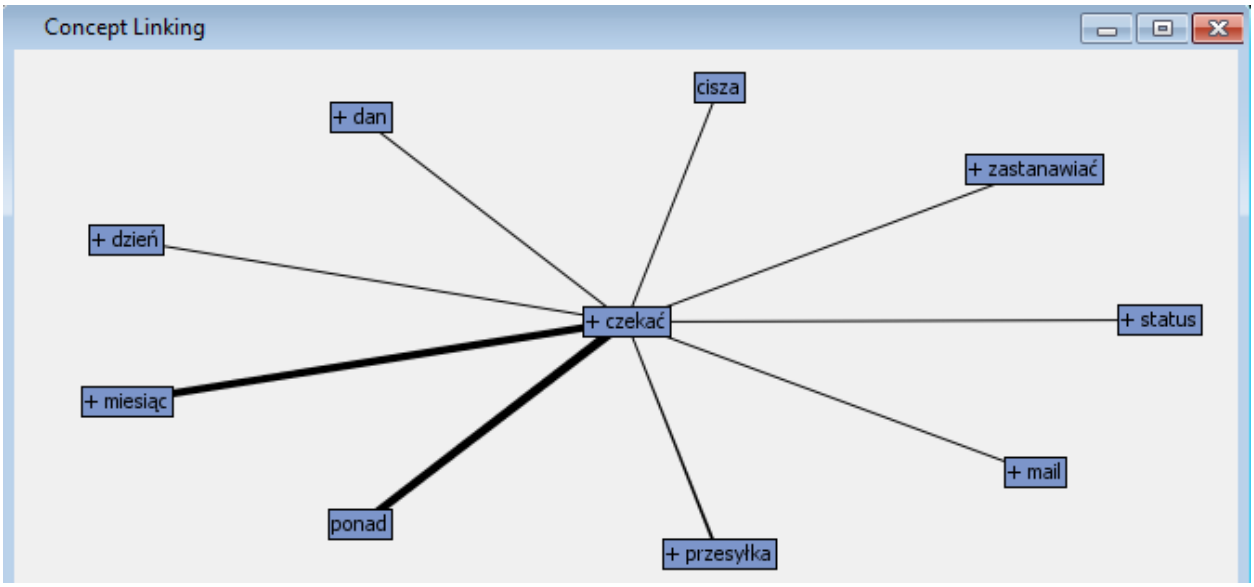
Figure 4. Clusters created by SAS Text Miner® 12.1 and their labels for the Customer Care text data

Beside clusters results of Text Filter can occur also very helpful in identifying keywords. Furthermore, Concept Links option provides information about which words are most frequently used along with chosen keyword. It is a very simple analysis but gives fast and easy insight. Below are the examples:



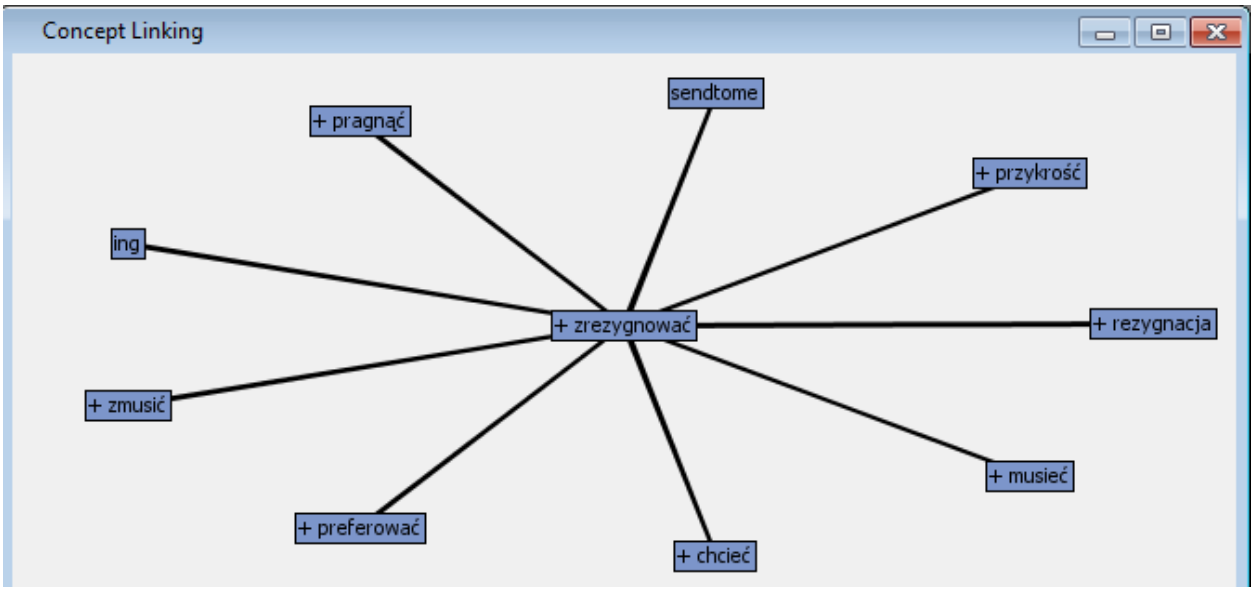
Terms mainly associated with service – nice, professional, helpful, agile, polite, helpful, competent

Figure 5. Concept linking with English counterparts for the Survey text data



Terms mainly associated with wait – over month, status, consider, mail, no answer

Figure 6. Concept linking with English counterparts for the Internet text data

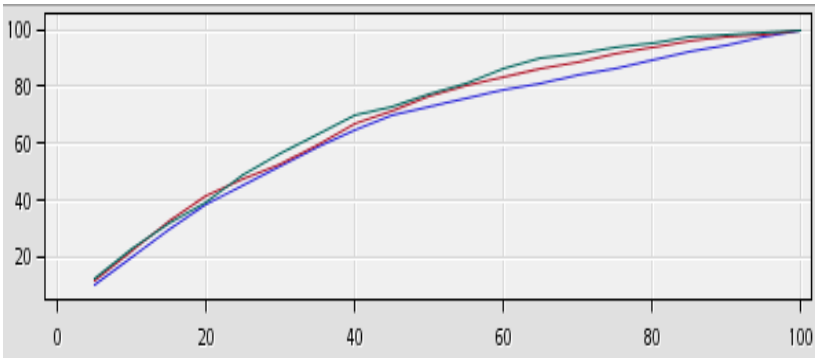


Terms mainly associated with resign – want, have to, resignation, prefer, regret

Figure 7. Concept linking with English counterparts for the Customer Care text data

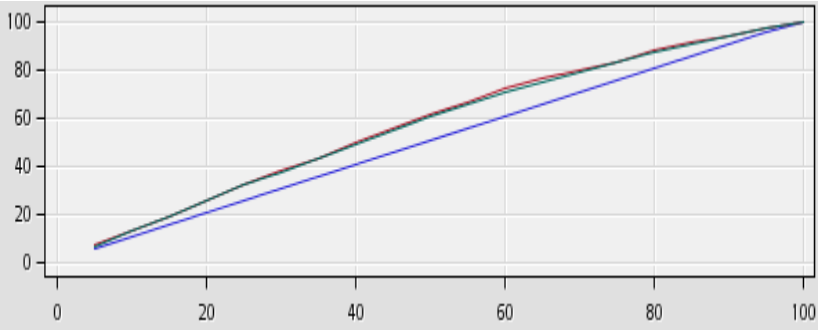
The method proposed for sentiment analysis resulted in models with misclassification rates around 25%. For Zendesk data Neural Network model is best, whereas for smaller in volume Internet data regression model is chosen. As in terms of sentiment analysis the model does not have to meet the requirement of being easy for interpretation, so Neural Network model is accepted. The models allow to assign sentiment to every analyzed document and, thus sentiment variables can be built to check how they relate with sales. However, clusters analysis already provides some intuition upon predominant sentiment for every one of three data sources, the sentiment analysis confirmed the

guesses. Therefore, survey contains generally positive opinions, antygroupon negative comments and customer care opinions are usually neutral – questions or refund requests. Below can be find comparison charts and stats for tested models:



Model	Missclassification rate
Neural Net	0.250
Regression	0.254
Decision tree	0.260

Figure 8. Sentiment model comparison for Customer Care data (Validation data)



Model	Missclassification rate
Neural Net	0.269
Regression	0.269
Decision tree	0.270

Figure 9. Sentiment model comparison for Internet data (Validation data)

The achieved above results can be next put together in order to prepare such Excel table for end user, as in the example:

Product ID	Source1_Neu Sentiment	Source1_Posit Sentiment	Source1_Negat Sentiment	Source1_Neu Cluster ID	Source1_Posit Cluster ID	Source1_Negat Cluster ID
1	5	5	5	1	2	3
2	6	6	6	3	2	1
3	7	7	7	6	5	4

Table 1. Example of table for end user containing data from Text Mining Analysis

The obtained sentiment variables are used in further sales analysis where explanatory variables are the numbers of opinions from different sources with different sentiment and dependent variable is a percentage realization of the chosen gross booking benchmark. Thus, the betas should be interpreted as percentages of the chosen benchmark. The simple regression model indicates that every sentiment variables is significant on each significance level. Surprisingly, betas for neutral and negative opinions from customer care source are positive (zen_sent0, zen_sent2), whereas beta for positive opinions is negative (zen_sent1). The possible explanation for this is that if product sells in huge volumes then there are higher chances that there will be more neutral or negative queries from clients about this particular product. So it can be stated that above some threshold the negative correlation changes to positive one. Positive customer care opinions most likely refer to products which sold worse than average and so the model shows negative influence of positive customer mails. Regarding, survey opinions betas signs seem to converge more with intuition and negative and neutral opinions have negative betas (surv_sent0, surv_sent2), whereas positive opinions have positive beta (surv_sent1). The results suggest that the relation between sales data and opinions may be not linear. The Intercept coefficient is not interpretable and for the same regression model but built on standardized data intercept is not significant.

Without category distinction R^2=0,3821

Parameter Estimates					
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr. > |t|
Intercept	1	-34.60169	1.38658	-24.95	<.0001
zen_sent0	1	9.30434	0.10506	88.56	<.0001
zen_sent1	1	-126.97942	3.33139	-38.12	<.0001
zen_sent2	1	8.81334	0.51099	17.25	<.0001
surv_sent0	1	-9.76213	1.64687	-5.93	<.0001
surv_sent1	1	2.52267	0.12523	20.14	<.0001
surv_sent2	1	-5.78663	0.66734	-8.67	<.0001

Figure 10. Statistics for simple regression model without category distinction

While simple regression model without product category distinction indicated that all variables are significant same model but grouped by category gives different results. The betas differ between groups, as well as variables significance. Below are the models for two different categories:

Category I R^2=0,2106

Parameter Estimates					
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr. > |t|
Intercept	1	-13.83791	1.26603	-10.93	<.0001
zen_sent0	1	11.74004	0.19949	58.85	<.0001
zen_sent1	1	-97.30383	16.65730	-5.84	<.0001
zen_sent2	1	-6.77627	0.70355	-9.63	<.0001
surv_sent0	1	-11.58138	3.53730	-3.27	0.0011
surv_sent1	1	0.46688	0.21642	2.16	0.0310
surv_sent2	1	-3.56106	0.88937	-4.00	<.0001

Figure 11. Statistics for simple regression model for Category I

Category II R^2=0,5353

Parameter Estimates					
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr. > |t|
Intercept	1	-61.46267	5.66818	-10.84	<.0001
zen_sent0	1	8.24597	0.21335	38.65	<.0001
zen_sent1	1	-112.99674	5.63825	-20.04	<.0001
zen_sent2	1	8.21471	1.05864	7.76	<.0001
surv_sent0	1	18.57244	26.39325	0.70	0.4817
surv_sent1	1	2.27702	2.02852	1.12	0.2617
surv_sent2	1	-9.21700	12.93752	-0.71	0.4762

Figure 12. Statistics for simple regression model for Category II

As already mentioned most likely the relation between sales and opinions is not linear, hence the simple decision tree model is generated. Additional, benefit of this model is the fact it provides rules that are very easy to comprehend.

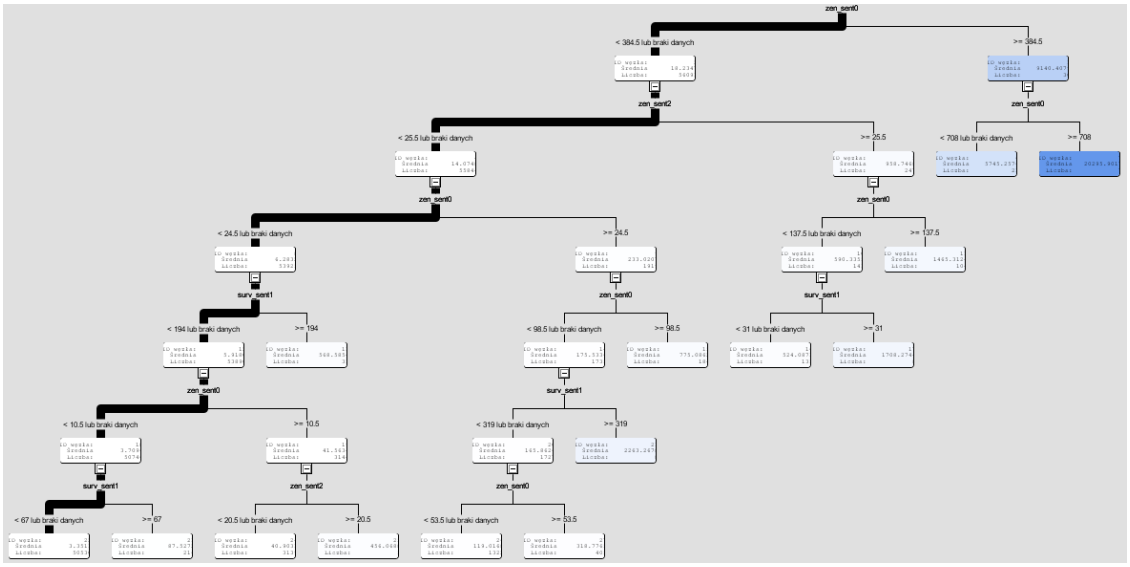


Figure 13. Simple decision tree model without category distinction

Examples of identified rules:

1) If Nbr zen_neutral>=385 then on average 9140% of benchmark gross booking
2) If Nbr zen_neutral<385 and Nbr zen_negative >=26 then on average 959% of benchmark gross booking
3) If Nbr zen_neutral<385 and Nbr zen_negative >=26 and Nbr zen_negative <138 and Nbr surv_positive>=31 then on average 1708% of benchmark gross booking

The results of decision tree confirm the assumption about non-linear relation between sales and opinions. Based on decision tree results it seems that zendesk opinions are more important for segmentation decisions. However, worth to be noted is the fact that there were 2,5 more opinions from customer care department than from surveys what may affects the results.

CONCLUSIONS

If company gathers text data or can identify sources of text data treating about its performance then it certainly should use them to increase understanding of its market and business. Using Text Mining Methods the company can find out about sentiment of the sources and their main topics/clusters. Companies offering different categories of product can check how opinions differ among them using obtained topics or clusters. Finally, achieved new variables from text analyses can be used to verify relation between opinions and sales results and if any relation exists then potentially enhance companies' sales models. The opinions gathered from Internet, that can be matched with appropriate sales, can be only used in general sales analysis, whereas in-house gathered data most likely have many different ids assigned to them and thus can be also applied to other analyses, for example customers purchase behavior.

REFERENCES

Albright, Russell. "Taming Text with the SVD." January 7, 2004: 72 paragraphs. Available www.sas.com/apps/whitepapers/whitepaper.jsp?code=SDM5.

Bergeron, P., Hiller, C. A. (2002). “Competitive intelligence”, in B. Cronin, Annual Review of Information Science and Technology, Medford, N.J.: Information Today, vol. 36, chapter8, 2002

Berry, M. J. A., Linoff, G. (2004). “Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management”, Wiley Computer Publishing, 2nd edition, 2004

Coussement, K. (2008). Employing SAS Text Miner Methodology to Become a Customer Genius in Customer Churn Prediction and Complaint E-mail Management. SAS Global Forum 2008. San Antonio: SAS Institute Inc.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T. and Harshman, R. (1990). Indexing by Latent Semantic Analysis. Journal of the American Society for Information Science 41 (6): 391-407.

Gao L., Chang, E. and Song, H. (2005). Powerful Tool to Expand Business Intelligence: Text Mining. World Academy of Science, Engineering and Technology

Ishikawa, H., et al. (1999). Document Warehousing Based on a Multimedia Database System, Proc. IEEE 15th Intl. Conference on Data Engineering, pp.168-173, 1999

Nassis, V., Rajugan, R., Dillon, T.S., Rahayu, J.W. (2004). Conceptual Design of XML Document Warehouses. In: Proceedings of the 6th International Conference Data Warehousing and Knowledge Discovery (DaWaK 2004), Zaragoza, Spain, Springer (2004) 1–14

Pramod.R, (2012), Search Engine Using SAS, MWSUG 2012 Conference Proceedings, Minneapolis, MSWUG

Salton, G. (1971). The SMART Retrieval System: Experiments in Automatic Document Processing. Prentice Hall, Englewood Cliffs, NJ.

Sanders, A., & DeVault, C. (2004). Using SAS® at SAS: The Mining of SAS Technical Support. SUGI 29.

SAS Institute Inc. (2012). SAS® Text Miner 12.1 Reference Help. Cary, NC: SAS Institute Inc.

Schulz. F How to import Twitter tweets in SAS DATA Step using OAuth 2 authentication style, SAS Voices 2013, Available at http://blogs.sas.com/content/sascom/2013/12/12/how-to-import-twitter-tweets-in-sas-data-step-using-oauth-2-authentication-style/

Singhal, A. (2001) Modern information retrieval: a brief overview. IEEE Data Engineering Bulletin, Special Issue on Text and Databases,24(4), Dec. 2001.

Sparck Jones, K. (1973). Index term weighting. Information Storage and Retrieval 9 (11): 619-633.

de Ville, B. (2001). “Microsoft Data Mining: Integrated Business Intelligence for e-Commerce and Knowledge Management”, Boston: Digital Press

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Rafał Wojdan
Warsaw School of Economics
Warsaw, Poland
Email: wojdanrafal@gmail.com

APPENDIX

1. Zendesk API macro to import data and put them into dataset.

```
/*Table to store the all retrieved data*/
```

```
proc sql;
create table zendesk.api_data (
id char(8),
   text char(25000)
  );
  quit;
  run;
/* s=number for start of the loop, e=number for end of the loop*/
%let s=1;
%let e=2;

%macro zendesk_api;
%DO i=&s %TO &e;
filename out ".../Zendesk/out&i..txt";
filename hout ".../Zendesk/hout&i..txt";
filename csvout ".../Zendesk/csvout&i..csv" encoding='utf-8';
run;

proc http
out=out
headerout=hOut
WEBPASSWORD="password"
WEBUSERNAME="login"
url="https://companyname.zendesk.com/api/v2/tickets.json?page=&i"
method="get"
ct="application/x-www-form-urlencoded";
run;
%include '...\proc_2.sas';
PROC IMPORT OUT= WORK.a
            DATAFILE= ".../Zendesk/csvout&i..csv"
            DBMS=CSV REPLACE;
     GETNAMES=YES;
       guessingrows=100;
     DATAROW=2;
RUN;

proc append base=zendesk.api_data data=work.a force;

proc sql;
drop table work.a;
quit;
run;
/*If there is the limit to number of communication with API per minut use
sleep function*/
data _null_;
call sleep(5,.1);
run;
%include '...\clear_groovy_memory.sas';
DM "log; clear; "; /*clear log*/
%end;
%mend zendesk_api;
```

%Include 1: Data conversion from JSON to CSV

```
proc groovy classpath="...\groovy-2.2.1\embeddable\groovy-all-
2.2.1.jar;...\opencsv-2.3\deploy\opencsv-2.3.jar";
   submit ".../Zendesk/out&i..txt" ".../Zendesk/csvout&i..csv";
```

```
      import groovy.json.*
      import au.com.bytecode.opencsv.CSVWriter

      def input = new File(args[0]).text
      def output = new JsonSlurper().parseText(input)
      def csvoutput = new FileOutputStream(args[1])
CSVWriter writer=new CSVWriter(new OutputStreamWriter(csvoutput, "UTF-8"));
        String[] header = new String[2];
      header[0] = "id";
      header[1] = "text";
writer.writeNext(header);
      output.tickets.each {
String[] content = new String[2];
content[0] = it.id.toString();
content[1] = it.text.replaceAll('\n|\r',' ').toString();
         writer.writeNext(content)
      }
      writer.close();
    endsubmit;
 quit;
```

%Include 2: Clear Groovy memory

```
proc groovy classpath="...\groovy-2.2.1\embeddable\groovy-all-
2.2.1.jar;...\opencsv-2.3\deploy\opencsv-2.3.jar";
clear;
quit;
```

2. Code to extract Internet data and map them to CSV file.

```
filename anty url "http://antygroupon.pl";
run;

data check;
infile anty dsd lrecl=5000 encoding='utf-8';
format record $1000.;
input record @@;
file "...\Desktop\anty.txt"; /*wrzucanie do wybranego pliku*/
put record;
run;

proc groovy classpath="...\Html_cleaner\htmlcleaner-2.7.jar;...\groovy-
2.2.1\embeddable\groovy-all-2.2.1.jar;...\opencsv-2.3\deploy\opencsv-
2.3.jar";
   submit ".../Desktop/anty.txt" ".../Desktop/anty_fin.csv";
      import groovy.json.*
      import au.com.bytecode.opencsv.CSVWriter
        import org.htmlcleaner.*

def input = new File(args[0]).text
def csvoutput = new FileOutputStream(args[1])
def cleaner = new HtmlCleaner()
def node = cleaner.clean(input)
def props = cleaner.getProperties()
def serializer = new PrettyXmlSerializer(props)
def xml = serializer.getXmlAsString(node)
def parsed = new XmlSlurper().parseText(xml)
```

```
      CSVWriter writer=new CSVWriter(new OutputStreamWriter(csvoutput, "UTF-
8"));
String[] header = new String[2];
      header[0] = "Subject";
      header[1] = "Date";
writer.writeNext(header);
      parsed.body.div.div.div.div.div.each {
            String[] content = new String[2];
            content[0] = it.h3.toString();
            content[1] = it.p[2].toString().replaceAll(/\n/,' ');
      writer.writeNext(content)
            }
 writer.close();
endsubmit;
quit;
run;
```

3. Code to find two most similar documents

```
data searched_text;
set data;
keep text_body;
run;
proc sql noprint;
select count(*) into :searched from searched_text;
quit;
%let searched=&searched;
/*Parse the searched document collection*/
proc tgparse data=searched_text
out=parseout1 key=key stemming=yes
tagging=no entities=no ng=std
stop=sashelp.engstop lang=Polish;
var text_body;
run;
/* Compute the weigths and assign filters */
proc tmutil data=parseout1 key=key;
control init release;
select reducef=4;
weight termwgt=entropy cellwgt=log;
output key=newKey KEEPONLY KEYFORMAT=TMSCORE;
run;
/*Joining terms from parsing to terms from weighting*/
proc sql noprint;
create table plstart as
select distinct term from key as a
right join newkey as b
on a.key=b.key
;
quit;
/* Tu mają być opisy merchanta czy deala z bazy groupon
plus komentarze z internetu (tytuł+komentarz w kolumnie body)
dokumenty groupona na początku pliku*/

data all_text;
set all_data;
_document_ = _n_;
keep text_body _document_;
```

```
run;
/*Parse the searched + searchable document collection*/
proc tgparse data=all_text
out=parseout2 key=key stemming=yes
tagging=no entities=no ng=std
start=plstart lang=Polish;
var text_body;
run;
/* Compute the weigths and assign filters */
proc tmutil data=parseOut2 key=key;
control init release;
select reducef=4;
weight termwgt=entropy cellwgt=log;
output out=spsvdIn key=newKey KEEPONLY KEYFORMAT=TMSCORE;
run;
proc sort data=parseout2 out=parseout_sort;
by _termnum_;
run;
/* Compute the SVD */
proc spsvd data=parseout_sort global=entropy
local=binary max_k=75 res=high p=50;
row _termnum_;
col _document_;
entry _count_;
output U=U S=S V=V;
run;
data distance.v1;
set distance.v;
length id $20;
id="doc"||compress(index);
run;
%let gr=%sysevalf(&searched+1);
data distance.v1b;
set distance.v1(firstobs=&gr);
run;
/* In this macro the whole collection is grouped in 50000 groups of
documents. The numbers instead of macro variable are left so that is it
easier to see how the partition works. Macro variable stop should equal the
result of equation all_texts/size of group, for example 200 000/50 000 =4 */
%macro similarity(stop);
proc sql noprint;
drop table sc1;
quit;
proc sql noprint;
drop table v1a;
quit;
proc sql noprint;
drop table matched_pairs;
quit;
%do i=1 %to &stop;
%if &i>1 %then %do;
%let a=%sysevalf(&i-1);
%let co=%sysevalf(&a*50000);
%let co1=%sysevalf(&co+1);
%let co2=%sysevalf(&co+49999);
data v1a;
set v1(firstobs=&co1 obs=&co2);
```

```
run;
%end;
%else %if &i=1 %then %do;
data v1a;
set v1(obs=50000);
run;
%end;
proc append base=v1a data=v1b;run;
proc distance data=v1a method=cosine
absent=0 out=distance.sc1;
   var ratio(col1--col75);
   id id;
run;
data matched_pairs;
set sc1 (firstobs=50001);
array sim(*) _numeric_;
idoferta=.;
do i=1 to 50000;
if sim(i)>0.7 and sim(i)<1 then do;
if &i=1 then do;
metric=sim(i);
idloop=&i;
id_pair=i;
output;
end;
else if &i>1 then do;
metric=sim(i);
idloop=&i;
id_pair=((&i-1)*50000)+i;
output;
end;
end;
end;
keep id id_pair metric idloop;
run;

proc append base=distance.matched_pair_fin data=distance.matched_pair
force;run;
%end;
%mend;
```

4. Macro exploiting Custom Search Google API

```
data adjusted;
set data;
subject=prxchange('s/\s+/%20/',-1,subject);
run;
option compress=yes;
proc sql noprint;
select count(*) into :loop from adjusted;
quit;
%let loop=&loop;
%macro api_google;
%DO i=1 %TO &loop;
proc sql noprint;
select subject into :subject from adjusted
```

```
where row=&i;
quit;
%let subject=&subject;
filename out ".../Desktop/out&i..txt";
filename hout ".../Desktop/hout&i..txt";
proc http
out=out
headerout=hOut
/* &cx and &q are API keywords not macro variables */
url="https://www.googleapis.com/customsearch/v1?key=xxxxxxxxxxxxxxx&cx=xxxxxx
xxxx&q=&subject."
method="get"
ct="application/x-www-form-urlencoded";
run;

%end;
%mend api_google;
```