

Predicting school funding requests that deserve an A+

putting theory into practice in a Kaggle competition

Research paper

Student: M.A. Kuijer

Supervisor: Dr. E. Haasdijk

Vrije Universiteit Amsterdam

Faculteit der Exacte Wetenschappen

Studierichting Bedrijfswiskunde en Informatica

De Boelelaan 1081a / 1081 HV Amsterdam



Preface

This research paper is a compulsory part of the Master's program Business Analytics at the VU University Amsterdam. Its requirements concern the inclusion of the different aspects of the study: Business, Mathematics and Informatics (former name) and the practical value of the research. Furthermore, completing the research within a predefined period is another valued requirement.

The goal of this research is to help DonorChoose.org identify (predict) school funding requests that deserve an A+. In other words, which funding requests are, based on certain criteria, the most exciting and most likely to raise money. This problem was made available at Kaggle.com as one of their many Data Mining competitions.

Based on thorough literature research, different models and techniques were selected in order to make the best possible predictions. Problems faced and decisions made while putting theory into practice are reported in this paper. These include: time constraints, computation- and software limitations as well as problem- and model-specific issues. This paper can be viewed as a step by step report based on my practical work. It tries to explain decisions made by providing theoretical background.

The intended audience ranges from anyone ever faced with problems concerning putting Data Mining theory into practice to students and teachers interested in specific algorithms like Random Forests and problems like imbalanced data.

My thanks go to my supervisor Dr. E. Haasdijk. This work would not have reached its present form without his many suggestions and enthusiasm.

Amsterdam, July 2014

Abstract

Kaggle.com is a website which allows users to participate in Data Mining competitions with actual business problems. The goal of the selected classifying problem was 'to identify school funding request that deserves an A+'. This competition was particularly challenging for a number of reasons. These include: highly imbalanced data, data containing text, data in relational format (multiple files), both numerical and categorical attributes and the attribute levels ranging from 2 to over 9000 levels.

A Random Forest (RF) model proved to accomplish the best results. The imbalanced nature of the data was solved by the use of a specific form of down-sampling. Also, some text mining techniques were incorporated in the final model.

The final model resulted in a 47th position in the competition (472 competing teams). Therefore, my top 50 goal in the Kaggle competition was achieved without extensive Data Mining/Machine Learning experience. This paper shows that a complex classifying problem can be solved with simple techniques and good understanding of the data. Many of the techniques and skills were acquired throughout the process (text mining, down-sampling techniques etc.).

Index

1. Introduction.....	5
2. Data.....	6
2.1 Data Files.....	6
2.2 Data Analysis.....	7
3. Literature.....	8
3.1. Imbalanced Data.....	8
3.1.1. Accuracy.....	9
3.1.2. ROC Curve.....	10
3.1.3. Up sampling, down sampling and cost-sensitive learning.....	11
3.1.4. Balanced Random Forest.....	12
3.2. Text mining / Text summarization.....	13
3.2.1. Pre-processing.....	14
<i>basics, document stemming, porters algorithm, stemming versus lemmatization, synonyms</i>	
3.1.2. Basic Summarization.....	16
3.1.3. Lexical Diversity.....	16
3.1.4. TF*IDF.....	17
3.1.5. Conclusion.....	18
3.3. Random Forest.....	18
3.3.1. Ensemble Learning: Random Forest.....	18
3.3.2. OOB: out of bag.....	18
3.3.3. Balanced Random Forest.....	20
3.3.4. Advantages.....	20
4. Method.....	21
4.1. Balanced Random Forest.....	21
4.2. Feature generation.....	21
4.2.1. Numerical and categorical.....	21
4.2.2. Text.....	22
4.2.3. Overlap approach.....	22
4.3. Variable inclusion.....	23
5. Results and Discussion.....	23
5.1. Validation.....	24
6. Conclusion.....	26
Appendix 1.....	27
1.1. Extra Information and Research question as posted on Kaggle.com.....	27
1.2. Advantages of competing in a Kaggle competition.....	28
Appendix 2.....	29
2.1. Data.....	29
2.2. File Descriptions.....	29
2.3. Exciting projects.....	29
2.4. Data Fields.....	30
2.5. Overview of data provided.....	33
Appendix 3.....	34
3.1. Kaggle leaderboard.....	34
3.2. Text mining graphs.....	35

1. Introduction

One way in which the United States (U.S.) differs from many western-European countries, is the gap between rich and poor. Recent studies have shown that in the U.S. the achievement gap between rich and poor children is widening. This development threatens to dilute education's levelling effects (New York Times, 2012). Educational inequity due to parents income has received far less attention than gaps in student accomplishment by race. This makes the problem of educational inequity and its causes an interesting research area and, more importantly, a research area with social relevance.

The website DonorsChoose.org, according to their website, *"engages the public in public schools by giving people a simple, accountable and personal way to address educational inequity"*. DonorsChoose.org does exactly what the name suggests. It allows donors to choose which school project deserves their financial support. Doing so improves the chances for success of the less advantaged children.

DonorsChoose.org is interested in identifying interesting projects early. This is particularly important because by identifying and recommending such projects early, they will improve funding outcomes, better the user experience, and help more students receive the materials they need to learn. However, helping DonorsChoose.org identify these projects may also generate insight into what factors drive people to give money to this kind of charity.

This classifying problem was selected as the annual KDD Cup at Kaggle.com. This website allows users to participate in Data Mining competitions with actual business problems. The 2014 KDD Cup's goal is 'to identify school funding request that deserves an A+'. More information about the challenge and the advantages of participating in such a competition can be found in the Appendix or on kaggle.com¹.

The objective of this research paper is to determine whether these exciting projects can be identified at the moment they are proposed. Furthermore, issues like imbalanced data, data containing both categorical and numerical attributes and text attributes are addressed. Strategies to deal with these are proposed and discussed. The theoretical parts of this paper cover approaches to the problem based solely on literature. Whereas the more practical chapters report on their outcomes.

To sum it all up, this paper attempts to answer a number of questions. First of all, the main question this paper tries to answer is *"how can DonorCHoose.org identify potentially exciting projects early?"* To answer this question, it is important to answer some more specific questions first. These include finding good approaches to deal with imbalanced data, text attributes to come up with a suitable Machine Learning model for the challenge.

Questions:

- How can DonorsChoose.org identify potentially exciting projects?
 - How to deal with an imbalanced dataset?
 - Can basic text-mining techniques be implemented to improve results?
 - What is the best Machine Learning model for this particular challenge?

¹ <http://www.kaggle.com/c/kdd-cup-2014-predicting-excitement-at-donors-choose>

2. Data

This chapter aims to explain the data provided and discusses its most eye-catching properties. Some assumptions are made in this chapter to simplify the problem given the time-window of one month. This chapter can be skipped but is useful when trying to understand certain decisions made throughout the process.

2.1 Data files

For the challenge, six .csv files were available. These ranged from information per project to a sample submission file. The table below shows information about the different files and their contents.

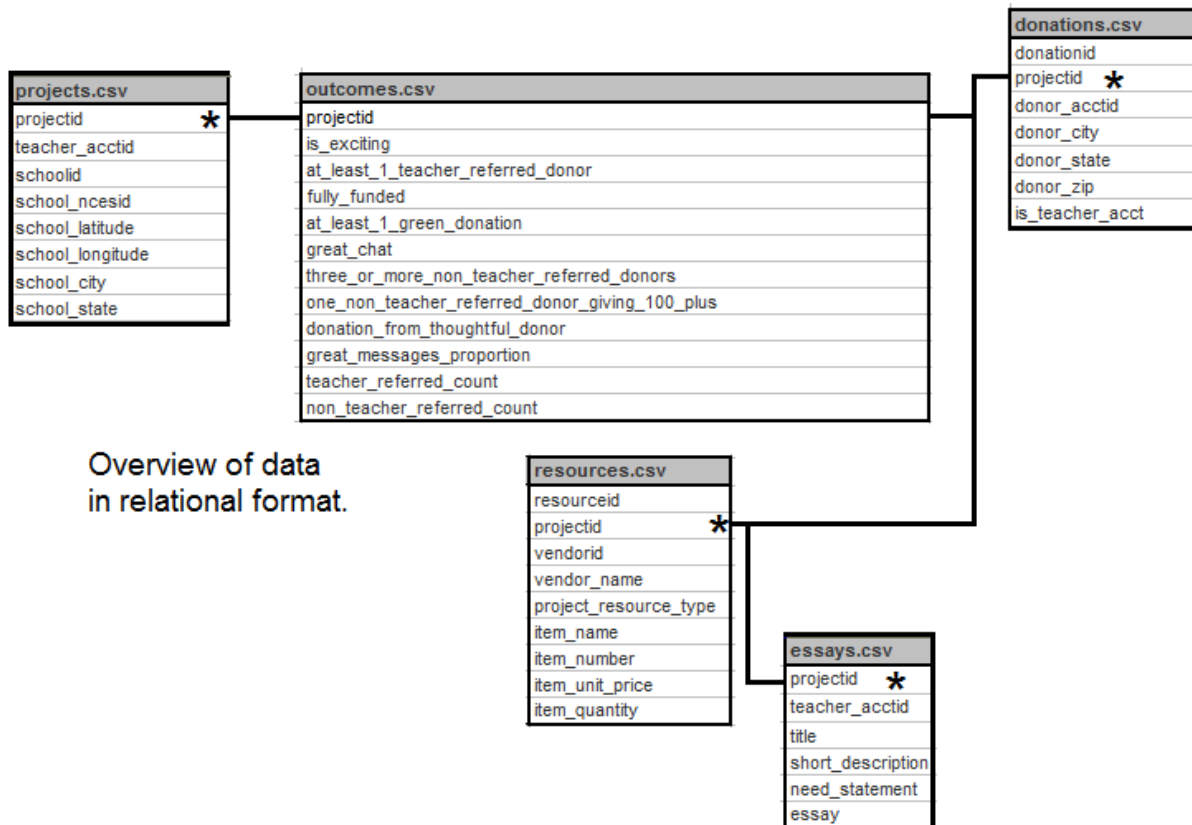


Table 2.1: Overview of the data and some of its attributes, in relational format. For a complete version (including all the attributes), please consult the appendix. Not included: the sample submission file. This file is not important when solving the problem.

These files contain both training- and test set data. The donations file, for example, is only available for the training set. This makes sense, because there is no knowledge about who will donate to what future projects. The different files contain almost anything there is to know about donations, resources needed, and the teachers making the requests. For a full list of the contents of the provided datasets, please consult Appendix I.

File	Number of instances (rows)	Number of attributes (columns)
• donations.csv	3.097.988	21
• essays.csv	664.098	6
• outcomes.csv	619.326	11
• projects.csv	664.098	35
• resources.csv	3.667.582	9
• sampleSubmission.csv	44.772	-

Table 2.2: Summary of the datasets. A total of 8757864 instances and 82 (not all unique) attributes

At this point, an important assumption was made. As was stated before, there is no knowledge about future donations. However, the donations file could be useful when ‘profiling’ teachers’ donating behavior. Because, for any donation, it is logged whether the person making the donation is a teacher or not. Again, because of the time constraint, it was decided to postpone the use of the donations.csv and focus on the other files first.

To simplify the problem, the resources file was also ignored. This file contains incredibly detailed information about the resources needed. However, these are summarized in more general attributes already present in projects.csv.

Therefore, this paper focuses on the files: projects.csv, essays.csv.

File Name
essays.csv
projects.csv

Table 2.3: The two important files. Obviously the outcomes file was also used because without this file, supervised learning would be impossible.

2.2 Data analysis

When observing the data, one may notice that many of the attributes are closely related. Examples of this are the obviously (cor)related regional attributes and school/teacher attributes. There are many more examples of these closely related attributes. This suggests the use of a model capable of dealing with correlated attributes. Either that, or paying close attention to feature selection and problems arising from using the correlated attributes. Another property of the data worth mentioning, is the fact that it is provided in relational format. When only using the projects and essays files however, this can be easily solved using merges by “project id”.

Another distinguishing aspect of the 2014 KDD cup is the highly imbalanced nature of the data. The number of exciting projects compared to the number of non-exciting projects is small. Before any data preparation or pre-processing steps were carried out, the fraction exciting projects was only 5.927%. Finally, the data is mixed (categorical, numerical, text) and there is some interesting overlap between training- and test set which is presented in the table below.

Attribute	Percentage of test set that can also be found in training set (overlap)
projectid	0%
teacher_acctid	55.19%
schoolid	89.6%
school_ncesid	90.2%

Table 2.4: *Overlap between training- and test set. These numbers were computed before any data preparation or pre-processing steps were carried out.*

Also, the first labeled 'exciting' project occurs on 14/04/2010 whereas the first proposed project was almost a decade before that. Therefore, all the entries before 14/04/2010 are much less interesting than the projects proposed after this date. These entries are also a significant number of years ago, and may not be as relevant anymore as more recent data. After removing these entries there are still more than enough entries remaining and due to this, the fraction of exciting projects almost doubles to around 9%.

Throughout the process, any missing values were replaced by their column means.

In short, the main (complicating) properties of the data are the *text attributes*, *imbalanced data* and *selecting a suitable model* based on these properties (and other properties of course).

3. Literature

Based on the observations of the previous chapter, this chapter covers the theoretical background of the main complicating properties of the data and discusses which models handle these properties best. The chapter assumes some basic knowledge of different Machine Learning models. However, any models proposed as a final model for the competition are explained thoroughly.

The chapter consists of three subchapters. The first of these provides theory on imbalanced data and discusses different ways to cope with this. Based on several scientific papers a suitable method to for imbalanced data problems is proposed for the Kaggle competition. The second subchapter provides theory on text mining in a similar manner. The final subchapter proposes, solely based on literature research, a suitable model (Random Forest) for the given problem.

3.1 Imbalanced Data

"A dataset is imbalanced if the classification categories are not approximately equally represented."
Chawla (2005)²

² Nitesh V. Chawla (2005), *Chapter 40 - DATA MINING FOR IMBALANCED DATASETS: AN OVERVIEW*, Department of Computer Science and Engineering, University of Notre Dame, IN 46530, USA, 2005, pp 853-867

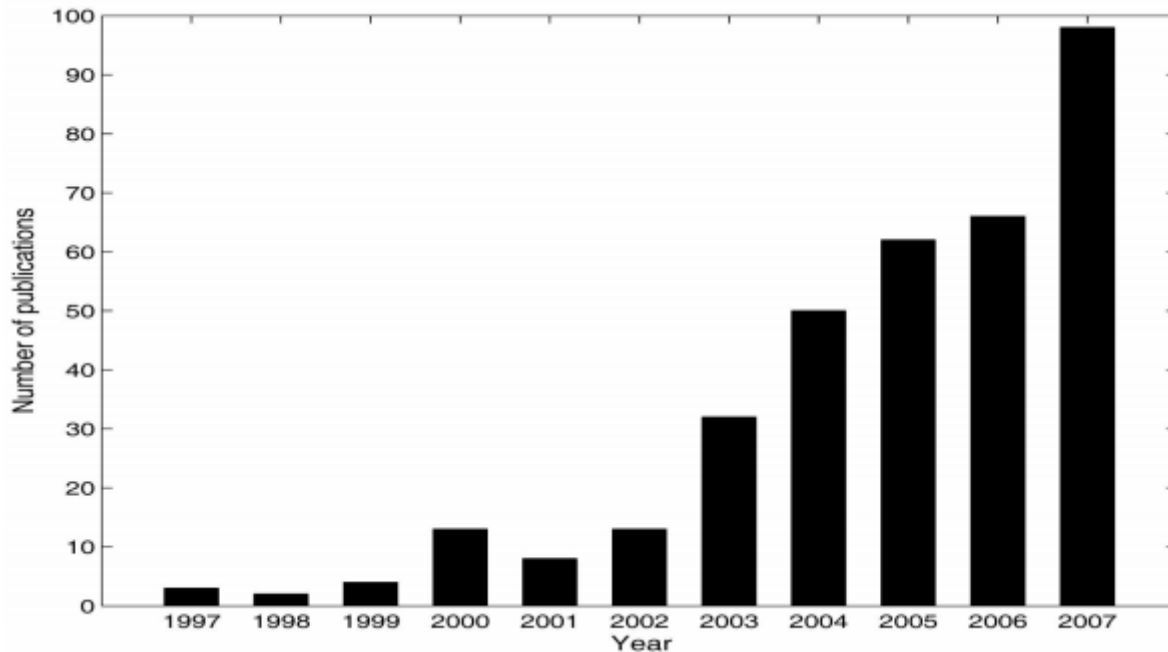


Figure 3.1: Number of publications on imbalanced learning (He & Garcia, 2009)³. This graph shows the increasing interest in improving imbalanced learning methods. This graph was included to emphasize on the relevance of this research area.

As mentioned earlier, the data sets for this classification task are highly imbalanced. The number of exciting projects compared to the number of less exciting projects is very small. This problem, known as the problem of imbalanced data, is very common in classification. The stumbling block, as Provost (2000)⁴ calls it, is the notion that an inductive learner produces a black box that acts as a labeling function. This causes problems when confronted with imbalanced class distributions. Provost (2000) continues by stating that most algorithms have two built-in assumptions that cause these problems:

1. the goal is maximizing accuracy, and
2. the classifier will operate on data drawn from the same distribution as the training data.

This results in, if no action is taken, unsatisfactory classifiers for unbalanced data sets. This is exactly what could happen when predicting exciting projects in the DonorChoose competition. Especially the first assumption influences the results because high accuracy is achieved by the trivial classifier that labels everything with the majority class. In other words, each new project is classified as not-exciting, this results in an impressive accuracy.

However, an algorithm predicting only no-instances is, without a doubt, useless. Therefore, this paragraph discusses multiple approaches to solve the imbalanced data problem. Also, a different evaluation method, other than accuracy, is discussed.

3.1.1 Accuracy

As the research in the Machine Learning area advanced, the limitation of the accuracy as the performance measure was soon recognized. ROC curves soon became a popular and more successful alternative (Fem et al., 2004 as cited in Chawla 2005).

Usually, a classifier is evaluated using a confusion matrix. This matrix contains information on the number of (mis)classified projects. There are four measures in the confusion matrix: True Negatives (TN) represent the number of negative examples correctly classified. The False Positives (FP) on the other hand, represent the number of negative examples incorrectly classified as positive. The False Negatives (FN) and True Positives (TP) respectively, represent the positive examples incorrectly classified as negative and the number of positive examples correctly classified (Chawla, 2005). The commonly used performance measure accuracy is calculated from these measures.

³ Haibo He, Member & Edwardo A. Garcia (2009), *Learning from Imbalanced Data*, *TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, VOL. 21, NO. 9, SEPTEMBER 2009.

⁴ Foster Provost (2000), *Machine Learning from Imbalanced Data Sets 101 - Extended Abstract*, , New York University.

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{FP} + \text{TN} + \text{FN})$$

Based on a confusion matrix:

		ACTUAL	
		Positive class	Negative class
PREDICTED	Positive class	True positive (TP)	False positive (FP)
	Negative class	False negative (FN)	True negative (TN)

Table 3.1: Confusion matrix, from McCarthy et al. (2005)⁵.

As stated earlier, accuracy is not a suitable measure for the imbalanced DonorChoose data. Another example often used to illustrate the imperfection of accuracy as a performance measure, is the classification of pixels in mammogram images (cancer detection). “A typical mammography dataset might contain 98% normal pixels and 2% abnormal pixels. A simple default strategy of guessing the majority class would give a predictive accuracy of 98%” (Chawla, 2005).

3.1.2 ROC curve

According to Chawla (2005) in “Data mining for imbalanced datasets: an overview”: “*The Receiver Operating Characteristic (ROC) curve is a standard technique for summarizing classifier performance over a range of tradeoffs between true positive and false positive error rates (Swets, 1988). The Area Under the Curve (AUC) is an accepted performance metric for a ROC curve (Bradley, 1997)*”.

Intuitively, an ROC curve makes sense. The x-axis represents the percentage of false positives (%FP = FP/[TN + FP]) and along the Y axis there is the percentage of true positives (%TP = TP/[TP + FN]). An ideal point on this curve would be (0,100), which means that all positive and negative examples are classified correctly. The curve illustrates the performance of a binary classifier as its discrimination threshold is varied.

⁵ Kate McCarthy, Bibi Zabar & Gary Weiss (2005), *Does Cost-Sensitive Learning Beat Sampling for Classifying Rare Classes?*, Fordham University, UBDM '05 Proceedings of the 1st international workshop on Utility-based data mining, Pages 69 - 77

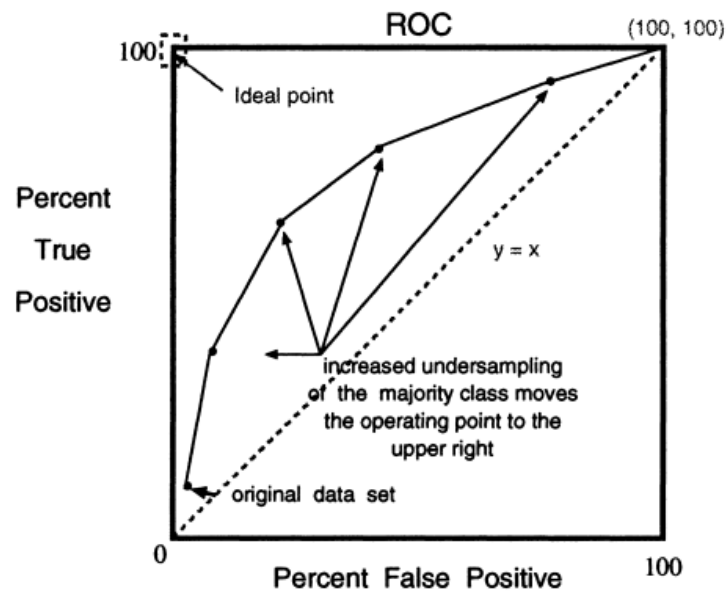


Figure 40.2. Illustration of Sweeping out an ROC Curve through under-sampling. Increased under-sampling of the majority (negative) class will move the performance from the lower left point to the upper right.

Figure 3.2: Source: Chawla, 2005: the line $y=x$ represents the scenario of randomly guessing the class.

One of the most commonly used performance measures for a ROC curve is the Area Under the Curve (AUC). While ROC is a two-dimensional representation of a model's performance, the AUC distills this information into a single scalar (Kaggle wiki⁶). The AUC is literally the area under the ROC curve. Therefore, In the scenario of randomly guessing the classes, the ROC curve will be a diagonal line stretching from (0,0) to (1,1) with an AUC of 0.5. The ideal point on the curve (0,100) on the other hand, results in an AUC of 1.

The Kaggle wiki on AUC confirms the advantages of AUC and also states a second advantage: *"Its main advantages over other evaluation methods, such as the simpler misclassification error, are:*

1. *It's insensitive to unbalanced datasets (datasets that have more installeds than not-installeds or vice versa).*
2. *For other evaluation methods, a user has to choose a cut-off point above which the target variable is part of the positive class (e.g. a logistic regression model returns any real number between 0 and 1 - the modeler might decide that predictions greater than 0.5 mean a positive class prediction while a prediction of less than 0.5 mean a negative class prediction). AUC evaluates entries at all cut-off points, giving better insight into how well the classifier is able to separate the two classes."*

Source: Kaggle wiki

Kaggle offers the possibility to make submissions throughout the duration of the competitions. For this competition, the evaluation method also used the AUC of a ROC curve (on 45% of the final data). The final ranking would be based on the AUC of the ROC curve on 100% of the data.

3.1.3 Up sampling, down sampling and cost-sensitive learning.

Since a highly-skewed class distribution usually causes predictions of the majority class only and therefore makes the natural distribution very often not the best distribution for learning a classifier (McCarthy et al., 2005), various techniques to deal with imbalanced data, have been introduced. Various resampling techniques are an example of this. These include: random over-sampling with replacement, random under-sampling, focused over-sampling, focused under-sampling, over-sampling with synthetic generation of new samples based on known information and combinations of the above techniques (Chawla et al., 2004b as cited in Chawla, 2005).

⁶ <https://www.kaggle.com/wiki/AreaUnderCurve>

McCarthy et al. (2005) generalize these methods to two basic methods for achieving a more balanced class distribution: up-sampling and down-sampling (also referred to as over-sampling and under-sampling). *"In this context, up-sampling replicates minority class examples and down-sampling discards majority examples"* (McCarthy et al., 2005). In other words, either some data is thrown away to balance the dataset, or data is added by copying from existing entries.

Another technique is cost-sensitive learning. Cost-sensitive learning is, according to McCarthy et al. (2005), the most direct method for dealing with highly skewed class distributions with unequal misclassification costs. When using cost-sensitive learning, a cost matrix represents the costs associated with the four outcomes in the confusion matrix. In other words: costs are assigned to the different outcomes. These are referred to as: CTP, CFP, CFN and CTN (C = cost). A close reader might notice there is an equivalency between cost-sensitive learning and sampling techniques. This subject, even though interesting, is beyond the scope of this paper.

It is not surprising that the discussed methods are not the only possibilities. There are many more techniques in the ever evolving field of machine learning. This paragraph however, focuses on up- and down-sampling in relation to cost sensitive learning because these methods are proven to deliver and are not too complex to implement. This is confirmed by McCarthy et al. (2005): *"because these [cost-sensitive learning, up- and down-sampling] are the only methods available to most practitioners"*. Based on the study by McCarthy et al. (2005) they are compared and their findings are used to determine which method is most suitable for the DonorChoose competition.

A clear disadvantage of down-sampling is that it discards potentially useful data. Up-sampling on the other hand increases the size of the training set, resulting in increased training time. Up-sampling is also sensitive to potential overfitting because it creates exact copies of existing entries. Therefore, it seems there are more disadvantages to sampling methods than to cost-sensitive learning. This is not entirely true, the disadvantage of down-sampling is indeed true for small datasets but not so much for bigger datasets. Since datasets have become bigger in size over the last couple of years, it has become common practice to down-sample balanced data as well (for computation time purposes). Therefore, discarding data has become much less of a problem than say, a decade ago. Luckily, the DonorChoose data is sufficiently large to attempt down-sampling. Up-sampling however, will increase training time and may cause overfitting.

For each of the three methods, finding ideal costs or sampling ratios is the stumbling block of solving an imbalanced data problem. In their study McCarthy et al. (2005) compared the three methods for a range of different datasets using a tree based learner. Before carrying out their research, their conjecture was that cost-sensitive learning should be preferred unless there are practical (computation time) or algorithm specific reasons not to. Their results however, showed otherwise. There is no clear "winner" when comparing the three techniques over a range of different datasets. Their datasets ranged from small in size to 20.000 rows. The DonorChoose data on the other hand, contains much more entries. This rules out up-sampling for practical (training time) reasons. Between down-sampling and cost-sensitive there is no clear underperformer. However, if down-sampling is used, there is the risk of throwing away potentially useful data.

3.1.4 Balanced Random Forest

It is worth researching whether there are ways to use down-sampling without the loss of potentially useful information. One could think of multiple samples and multiple runs of algorithms (with or without replacement). However, these runs will be time consuming which was the reason to rule out up-sampling in the first place. Luckily, there are Machine Learning algorithms that generate multiple bootstrap samples of training data in their training process. Normally speaking, when learning imbalanced data, there is a significant probability that a bootstrap sample contains few or even none of the minority class. It gets interesting when these bootstrap samples are modified. Chen et. al. (2005)⁷ describe the process of doing so and come up with a Balanced Random Forest (BRF) algorithm which modifies the bootstrap samples. They state: *"As recent research shows (e.g., Ling & Li (1998), Kubat & Matwin (1997), Drummond & Holte (2003)), for the tree classifier, artificially making class priors equal either by down-sampling the majority class or over-sampling the minority class is usually more effective with respect to a given performance measurement, and that downsampling seems to have an edge over over-sampling. However, down-sampling the majority class may result in*

⁷ Chao Chen, Andy Liaw, Leo Breiman (2004), *Using Random Forest to Learn Imbalanced Data*.

loss of information, as a large part of the majority class is not used. Random forest inspired us to ensemble trees induced from balanced down-sampled data.”

The distinguishing factor in their algorithm works as follows: *“For each iteration in random forest, draw a bootstrap sample from the minority class. Randomly draw the same number of cases, with replacement, from the majority class.”* (Chen et. al, 2005).

In their study, Chen et. al (2005) focus on using Random Forest to learn imbalanced data. The BRF is compared to a Weighted Random Forest (WRF, based on cost-sensitive learning). Thus, down-sampling (2.0) is again compared to cost-sensitive learning. This research also concludes that there is no clear winner between a down-sampling approach and a cost-sensitive approach (using Random Forest). It is concluded though, that these approaches have performance superior to most existing techniques that were researched by Chen et al. (2005). The study also establishes that BRF is computationally more efficient with large imbalanced data, since each tree only uses a small portion of the training set to grow, while WRF needs to use the entire training set. It is also concluded that WRF is possibly more vulnerable to noise than BRF.

To sum it all up, there are three widely used techniques to deal with imbalanced data: cost-sensitive learning, up- and down-sampling. Imbalanced data can not be evaluated using accuracy as a performance measure. A widely used alternative is the AUC of the ROC curve. This alternative is also used to calculate the ranking for the Kaggle competition.

Between cost-sensitive and sampling techniques, recent research has not been able to distinguish clearly which method has an advantage over the others. However, discarding potentially useful data in small datasets when down-sampling can be harmful. On the other hand, overfitting is a possible result of up-sampling due to exact copies of data entries. Algorithms like BRF do not discard data and are computationally more efficient than their cost sensitive alternative (WRF). Based on this research (Chen et al., 2005) a BRF or any other algorithm that uses bootstrap samples could be a candidate for the DonorChoose competition.

3.2 Text Mining / Text summarization

“Text Mining is the process of analyzing text to extract information that is useful for particular purposes”. (Witten et al., 2011)⁸.

Broadly interpreted, all natural language processing comes under the ambit of text mining.” (Witten et al., 2011).

The provided data for the Kaggle competition contains a number of text attributes that can not be included without converting them to a more acceptable form. This is where text mining comes in. When following the definition of data mining by Witten et al. (2011), who formulated that data mining *“is the extraction of implicit, previously unknown, and potentially useful information from data”*, one could argue whether text mining is a form of data mining. Because with text mining, the information to be extracted is clearly and explicitly stated in the text. The problem is that the information is not couched in a manner that is amenable to automatic processing (Witten et al., 2011). This is exactly what text mining tries to achieve, converting texts to a form which is suitable for consumption by computers. There is no real consensus about what text mining covers. A general definition is that all natural language processing (NLP) tasks are a part of text mining. In practice, text mining often comes down to summarizing possibly interesting properties from a large body of text (text summarization). This chapter adapts the definition of text mining by Witten et al. (2011) in their book *“Data Mining, Practical Machine Learning Tools and Techniques”*. Other definitions would require the name of this chapter to be changed to Information Retrieval (IR) or Natural Language Processing (NLP).

Since the text attributes in the problem at hand represent only a small fraction of the total number of attributes (numerical, categorical, text), this chapter focuses on a range of basic techniques that can be useful when a classifying problem contains a fraction of text attributes that in their current form can not be fed to a computer (or included in a model).

⁸ Ian H. Witten, Eibe Frank, & Mark A. Hall (2011), *Data Mining, Practical Machine Learning Tools and Techniques*, 3rd edition, 2011

3.2.1 Pre-processing

3.2.1.1 Basics

When viewing text mining as a number of techniques to convert text into a summarization of interesting properties which are suitable for consumption by computers, a pre-processing step seems a logical starting point. Punctuation, numbers, capital letters and function words (also known as stop words) sabotage a successful application of summarization techniques (not always!). Stop words are words that are usually filtered out before processing natural language data (text). There is no definite list of the words, but some of the most commonly known are: “the”, “is”, “at”, “which” and “on”. Usually, these words have no significant value. Removing stop words can improve performance of search engines (or machine learning algorithms) dramatically. However, problems arise when search terms include stop words, like: “The Who”, “Take That” or “The The”.

In particular cases, removing capital letters or numbers could result in a loss of information. For example, when analyzing twitter feeds, a tweet with much use of capitals may or may not be likely to be retweeted. Therefore, it is important to be careful during the pre-processing.

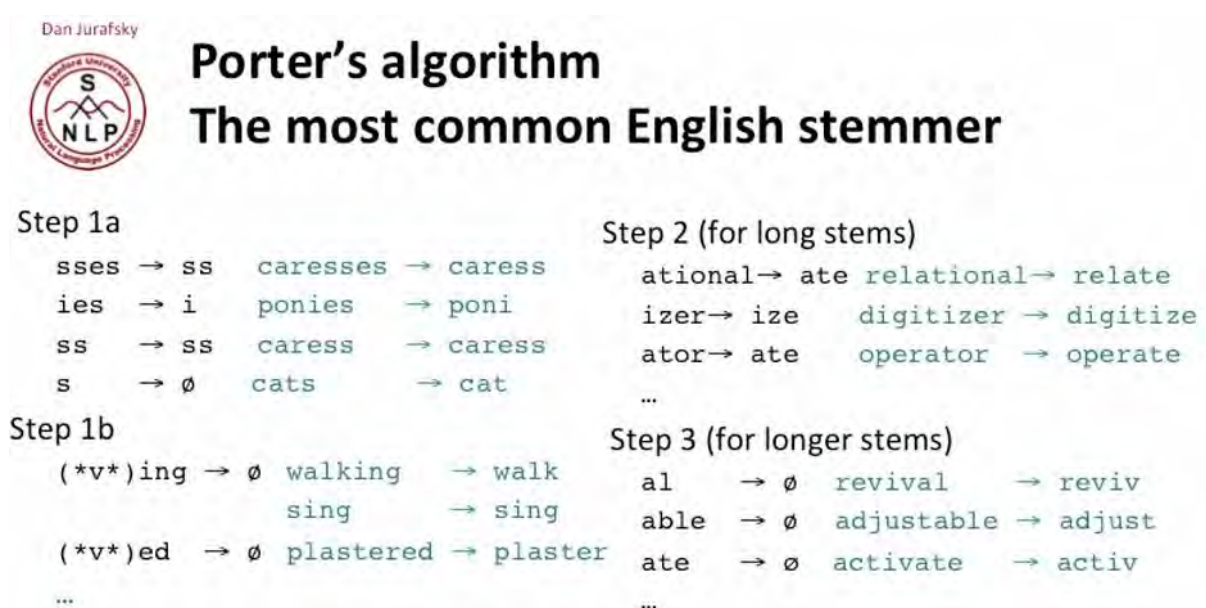
3.2.1.2 Document stemming

Another, commonly used pre-processing method is document stemming. The main purpose of stemming is to reduce different grammatical forms of a word like its noun, adjective, verb and adverb to its root form. Because the meaning of different grammatical forms is usually the same, each word form has to be identified with its root (or base) form. The most common stemming algorithms remove any attached suffixes and prefixes (affixes). It is important to realize that stemming is language dependent. Also, most of the stemming experiments done so far are for English and other west European languages (Jivani, 2011)⁹.


3.2.1.3 Porter's algorithm

Even though proposed back in 1980, Porters stemmer is still one of the most popular stemming methods. *“It is based on the idea that the suffixes in the English language (approximately 1200) are mostly made up of a combination of smaller and simpler suffixes”* (Jivani, 2011).

It is a rule-based algorithm consisting of 5 steps, and within each step, rules (60 in total) are applied until a passing condition is fulfilled. When the condition is fulfilled, the word is changed accordingly and the next step is performed. The final stem at the end of the last step is returned. Porter also designed a framework, called Snowball, which allows programmers to develop their own stemmers for other character sets or languages (Jivani, 2011).



Dan Jurafsky

 **Porter's algorithm**
The most common English stemmer

Step 1a

sses → ss	caresses → caress
ies → i	ponies → poni
ss → ss	caress → caress
s → ∅	cats → cat

Step 1b

(*v*)ing → ∅	walking → walk
	sing → sing
(*v*)ed → ∅	plastered → plaster
...	

Step 2 (for long stems)

ational → ate	relational → relate
izer → ize	digitizer → digitize
ator → ate	operator → operate
...	

Step 3 (for longer stems)

al → ∅	revival → reviv
able → ∅	adjustable → adjust
ate → ∅	activate → activ
...	

⁹ Anjali Ganesh Jivani (2011), *A Comparative Study of Stemming Algorithms*, Department of Computer Science & Engineering - The Maharaja Sayajirao University of Baroda - Vadodara, Gujarat, India

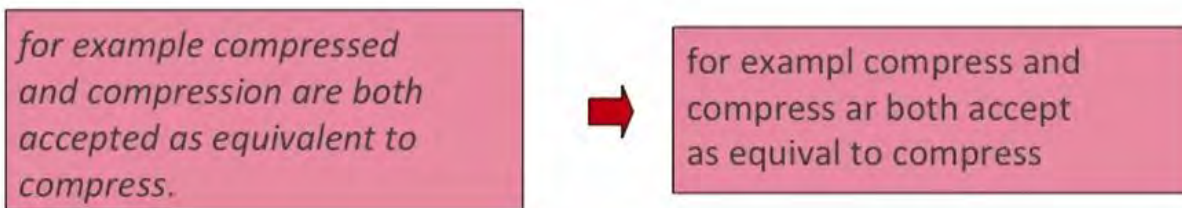


Figure 3.3: A sample of the rules from step 1-3 in Porters algorithm and an example of stemming using the algorithm. All steps are performed after which a final stem is returned after step 5. (*v*) = vowel
 Source: NLP Stanford video lecture Dan Jurafsky¹⁰.

Porters stemmer may be one of the most popular stemmers, but not the only one. Other popular stemmers are: Krovetz-, Xerox-, N-Gram-, HNN-, Dawson-, Paice / Husk -, YASS- and Lovins stemmer. Jivani (2011) conducted a comparative analysis of 9 different stemmers. In this research Porters algorithm was praised for producing the best output, having a small error rate and creating a language independent approach to stemming (Snowball). However, there are also some disadvantages to this algorithm. One being that it is time consuming. Each word goes through all the 5 steps and 60 rules. A second disadvantage is that the stems produced are not always real words. For machine learning purposes this does not have to be a problem. However, if more advanced NLP techniques are to be applied, like semantic analyses, this could be problematic.

Once stemmed, an analysis of a document’s lexical diversity or a term frequency matrix is more reliable and hopefully more valuable in a machine learning model. Therefore, stemming is a very useful technique in NLP and equivalently in text mining.

3.2.1.4 Stemming versus lemmatization.

A similar technique to stemming is lemmatization. The idea behind both techniques is the same: reducing a word variant to its ‘stem’ in stemming and ‘lemma’ in lemmatizing. Stemming however ignores the context of the word. In other words, it is a simpler technique to apply. *“In stemming, conversion of morphological forms of a word to its stem is done assuming each one is semantically related”* (Jivani, 2011). In practice this works quite well, for languages with relatively simple morphology. However, this often results in non-existent words.

In contrast, lemmatizing deals with the complex process of first understanding the context. Eventually, the word is reduced to its ‘lemma’.

<p>Stemming introduction, introducing, introduces – introduc gone, going, goes – go</p>	<p>Lemmatizing: introduction, introducing, introduces – <u>introduce</u> gone, going, goes, <u>went</u> – go</p>
--------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------

Table 3.2: Difference between stemming and lemmatizing (Jivani 2011). Lemmatizing results in existing words and can even convert past simples (‘went’) to its root.

3.2.1.5 Synonyms

Though beyond the scope of this paper. Another interesting pre-processing option in NLP is synonym detection. There are many thesauri available with synonyms that can be easily implemented. However, according to Wei et al. (2010)¹¹ most of these synonyms have the same or nearly the same meaning only in some senses. Synonym discovery is highly context sensitive. Replacing all synonym occurrences in for example Web search can therefore be tricky. It could ultimately lead to search intent drifting. When ‘predicting funding requests that deserve an A+’ and focusing on the text attributes, synonym replacement can be just as difficult. Ultimately, this can lead to valuable information being lost. Wei et al. (2010) explain this with the synonyms “baby” and “infant” which are treated as synonyms in many thesauri, but “Santa Baby” has nothing to do with an infant. “Santa baby” is actually

¹⁰ Natural Language Processing video lectures from Stanford University on coursera.org by Dan Jurafsky and Christopher Manning <https://class.coursera.org/nlp/lecture>

¹¹ Xing Wei, Fuchun Peng, Huishin Tseng, Yumoa Lu, Xuerui Wang, Benoit Dumoulin, *Search with Synonyms: Problems and Solutions*, - Yahoo! Labs, 701 First Avenue, Sunnyvale, California, USA, 94089, International Conference on Computational Linguistics 2010

the title of a song, and the meaning of “baby” in this entity is different than the usual meaning of “infant”. Therefore, synonym discovery is highly context sensitive. The context can not only limit the use of synonyms (“baby” and “infant”). It can also broaden the definition. Examples of these are “dress” and “attire” or “free” and “download”. In some contexts these word combinations are practically synonyms. In a Web search query for example, “free cd rewriter” may carry the same intent as “download cd rewriter” (Wei et al., 2010). Lastly, especially on the web, many new synonyms develop over time. “Mp3” and “Mpeg3” were not synonyms 20 years ago. However, manually updating synonym lists is extremely expensive. Therefore, when applying synonym detection methods, it is important to be aware where the texts come from. When analyzing books for example, synonym detection may be difficult when analyzing “The Merchant of Venice” and “The Da Vinci code”. Written in different times synonym detection could be difficult to apply.

There are however, automatic synonym discovery systems. In practice, they turn out to be a bit too advanced for the job at hand (the Kaggle competition). Based on this small research about the current state of synonym detection, it is not advisable to incorporate such advanced methods for such small tasks. The fraction of text attributes is relatively small compared to the other types of attributes. Also, when we are analyzing things like lexical diversity replacing synonym occurrences makes no sense. Without synonym detection and only using basic summarization techniques a lot of information about a text can be stored in numerical and categorical attributes.

3.2.2 Basic Summarization

This paragraph describes some basic summarization techniques and gives intuitive examples of their usefulness. These are by far not the only summarizing features. This paragraph however, is intended to give an idea of some commonly used basic summarization techniques. With some surprisingly easy features, one can (indirectly) include entire texts into a machine learning model.

The goal of summarization is to generate categorical or numerical features that give information about the text at hand. A good example is the problem of assigning a category to books in a library. Lets say there are four categories: short stories, children’s books, romantic novels and adult literature. Using the text in all the books available one could come up with measures like average word length, number of words, author, term frequency and lexical diversity. It makes sense that knowing the author of a book will help classifying the book. The number of words in the story might help assigning a book to the short story category. On the other hand, lexical diversity as well as average word length can help distinguishing between children’s books and adult literature. Lastly, a term frequency matrix can help recognizing which books belong to the romantic novel category. These books typically contain words like “love”, “romance”, “relationship” etc. This example illustrates how some simple features can enable the use of machine learning models based on text documents.

Often used features are: average word/sentence/paragraph length and number of words/sentences/paragraphs in the text. Lexical diversity, similarity, term frequency and latent semantic analysis are a little more advanced techniques and are therefore described in the paragraphs below.

3.2.3 Lexical Diversity

Many studies have shown that, in the field of automatic essay scoring for second language (L2) learners of English, essay length and lexical diversity are strongly correlated with essay ratings (Mellor, 2011)¹². This suggests that both essay length and lexical diversity are not only often used in this field but this also makes them interesting attributes in a machine learning model. Lexical diversity is a measure of the number of different words used in a text (Johansson, 2008)¹³. The measure is easy to implement and practical to apply in computer analyses of large data corpora. In other words, *“the more varied a vocabulary a text possesses, the higher the lexical diversity”* (Johansson, 2008). After all numbers, capitals and punctuation is removed in pre-processing steps and the words have been stemmed, this ratio becomes an even more powerful measure. Traditionally, the lexical diversity

¹² Andrew MELLOR (2011), *Essay Length, Lexical Diversity and Automatic Essay Scoring*, Department of Media Science, Faculty of Information Science and Technology, Memoirs of the Osaka Institute of Technology, Series B, Vol. 55. No. 2(2011) pp.1-14

¹³ Victoria Johansson (2008), *Lexical diversity and lexical density in speech and writing: a developmental perspective*, Lund University, Dept. of Linguistics and Phonetics, Working Papers 53 (2008), 61-79

measure is the ratio of different words (types) to the total number of words (tokens). This is the so-called type-token ratio, also known as TTR. When text size increases, TTR values decrease and vice versa. This has to do with reusing several function words in order to produce one new (lexical) word. A consequence of this is that TTR is only useful when comparing texts of equal length, because TTR penalizes longer essays (Mellor, 2011). Many alternatives have been proposed, the vast majority of them are adjusted alternatives to standard TTR trying to get around the effect of essay length. However, for the Kaggle competition, all text fields are limited to a maximum number of characters which allows the use of standard TTR.

$$\text{Lexical Diversity} = \frac{\text{\# unique words in essay}}{\text{\# words in essay}}$$

Formula 3.1: Lexical diversity of an essay. When the lexical diversity is equal to 50%, this means that every other word is a unique word.

3.2.4 TF*IDF

The TF*IDF term weight algorithm is widely applied into language models to build NLP Systems and is the most widely used weight algorithm nowadays (Xia & Chai, 2011)¹⁴. It consists of two parts. TF, the term frequency, also called Local Term Weight because it does not take other documents in consideration, is defined as the number of times a term (word) in question occurs in a document. For documents, the frequency for each term may vary greatly. Therefore, frequency is an important attribute of term to discriminate itself from other terms (Xia & Chai, 2011).

“IDF (inverse document frequency) assumes that the importance of a term relative to a document is inversely proportional to the frequency of occurrence of this term in all the documents” (Zhang et al., 2010)¹⁵. In other words; IDF is based on counting the number of documents in the collection being searched that are indexed by the term (Xia & Chai, 2011). IDF is also known as Global Term Weight, because it takes all the documents into consideration.

TF*IDF was evolved from IDF with the heuristic intuition that a term which occurs in many documents is not a good discriminator (Zhang et al., 2010). TF*IDF is therefore used as an indicator of the importance of a term in representing a document. (Xia & Chai, 2011).

$$w_{i,j} = tf_{i,j} \times \log \left(\frac{N}{df_i} \right)$$

Formula 3.2: classical formula for TF*IDF weighting (Zhang et al., 2010)

In the classical formula of TF*IDF, as depicted above, $w_{i,j}$ represents the weight for term i in document j , N is the number of documents in the collection, $tf_{i,j}$ is the term frequency of term i in document j and df_i is the document frequency of term i in the collection. According to Robertson (2004) as cited in Zhang et al. (2011) “the basic idea of TF*IDF is from the theory of language modelling that the terms in a given document can be divided into two categories: those words with eliteness and those words without eliteness”. In other words, TF*IDF is a measure to show the importance of a term in the document collection.

¹⁴ Tian Xia, Yanmei Chai (2011), *An improvement to TF-IDF: Term Distribution based Term Weight Algorithm*, JOURNAL OF SOFTWARE, VOL. 6, NO. 3, MARCH 2011

¹⁵ Wen Zhang, Taketoshi Yoshida, Xijin Tang (2010), *A comparative study of TF*IDF, LSI and multi-words for text classification*, Expert Systems with Applications, Elsevier 2010

As with almost any algorithm in the field of text mining, there are some drawbacks to this algorithm as well. Some of the criticisms are that it is too 'ad hoc' because TF*IDF is not directly derived from a mathematical model. Another problem is the size of the feature set (dimensionality). The size of a document term matrix (a sparse matrix) is based on the size of the vocabulary across the entire dataset. This brings about a huge computation on weighting all these terms (Christopher & Hinrich (2001) as cited in Zhang et al. (2011). Xia & Chai (2011) also state that even though stop words are often removed the problem of empty and function terms, including conjunctive, preposition, some adverbs, auxiliary term, modal particles that are usually present with a high frequency cannot be completely resolved. Despite these drawbacks, TF*IDF remains the most popular weight algorithm nowadays.

In conclusion, TF*IDF is another technique used to transform text into numerical vectors, i.e. text representation. For the Kaggle competition however, with the few text attributes present, one might need to consider the trade-off between model improvement and computation time.

3.2.5 Conclusion

Since one of the main themes in text mining is text representation, which is fundamental and indispensable for text-based intelligent information processing (Zhang et al. 2010), this chapter focused on text representation techniques. There is no real consensus to what domains the different aspects of NLP actually belong to. However, whether transforming text into numerical vectors falls under text mining or a different field, the problems to be solved remain the same. This chapter attempted to explain some popular and simple techniques to help extract information from the text attributes in the Kaggle competition. Using these techniques results in a number of new features that can be easily implemented in a Machine Learning model (Random Forest, Support Vector Machine, Neural Network, Decision Tree etc.). Whether these actually improve the model is to be determined by trial and error.

3.3 Random Forest

Before justifying the selected final model, Random Forest, This chapter starts with some preliminaries about Random Forest. Doing so, basic knowledge about constructing single classification trees is assumed.

3.3.1 Ensemble Learning: Random Forest

Ensemble learning is a popular technique within supervised learning to obtain better predictive performance than with constituent algorithms. "Ensemble methods are learning algorithms that construct a set of classifiers and then classify new data points by taking a (weighted) vote of their predictions" (Dietterich, 2000)¹⁶. Ensembles can often perform better than any single classifier (Dietterich, 2000). There are many different and commonly used ensemble learning algorithms like bagging, boosting, AdaBoost and Stacked Generalization. Deciding between these algorithms can often be dictated by trial and error and applicability (e.g. computation time).

Random Forest is an ensemble learning method for classification (and regression). It operates by constructing a multitude of decision trees. In order to classify a new object from an input vector, the input vector passes through each of the trees in the forest. This results in a classification by each of the trees. Like with any ensemble learning method, a data point is classified by a vote of the classifications. Since each tree gives a classification, they are said to "vote" for a particular class

¹⁶ Thomas G. Dietterich (2000), *Ensemble Methods in Machine Learning*, Multiple Classifier Systems, Lecture Notes in Computer Science Colume 1857, 2000, pp 1-15.

(Breiman, 2001)¹⁷. Obviously, the classification having the most votes (over all the trees in the forest) is selected. Random Forests (like any forest) can consist of thousands of trees. Each tree is grown according to the following scheme:

Lets assume the number of cases in the training set is N and there are M input variables. Now for every new tree:

1. With replacement, sample N cases at random from the training set. This sample will be the training set for the single tree.
2. Specify a number $m < M$ such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. This value m is held constant during the forest growing.
3. There is no pruning involved. Each tree is grown to the largest extent possible.

Table 3.3: Tree growing scheme for a tree in a Random Forest based on website by Breiman & Cutler¹⁸.

Due to sampling with replacement in step 1, all the N sampled cases can have duplicate data records and can be missing several data records from the original dataset. This is called bagging, the “out of the bag” values (or examples) turn out to be an important measure to decrease forest error rate.

3.3.2 OOB: out of bag

Breiman (2001) shows that the forest error rate depends on two things. First of all, there is the correlation between any two trees in the forest. Increasing the correlation increases the error rate. Secondly, the strength of each individual tree obviously influences the error rate. A tree with a low error rate is a strong classifier. Increasing the strength of individual trees decreases forest error rate. When m is reduced, both the correlation and the strength is reduced. Consequently, increasing m will increase correlation and strength. There is clearly a trade-off between maximizing strength and minimizing correlation between any two trees. A value m within an optimal range, usually quite wide, can be found using the so-called Out Of Bag (OOB) error rate (Breiman & Cutler). According to this website, m is the only adjustable parameter (apart from number of trees) to which Random Forests are somewhat sensitive.

When the training set for the current tree is drawn with replacement, about one-third of the instances are left out (Breiman, 2001). This is due to sampling with replacement. The one-third of instances that are left out, are not used in the construction of the k th tree. *“This OOB data is used to get a running unbiased estimate of the classification error as trees are added to the forest. It is also used to get estimates of variable importance”* (Breiman & Cutler). This works as follows, each case is left out in the construction of the k th passes through the k th tree to get a classification. Doing so, a test set classification is obtained for each case in about one-third of the trees. When we take j to be the class that got the most votes every time case n was OOB. Then the OOB error estimate is equal to *“the proportion of times that j is not equal to the true class of n averaged over all cases”* (Breiman & Cutler). The out-of-bag estimates are unbiased and because of the above, Random Forests do not require cross validation or a separate test set to get an unbiased estimate of the test set error (Breiman, 2001). *“Strength and correlation can also be estimated using out-of-bag methods. This*

¹⁷ Leo Breiman (2001), *RANDOM FORESTS*, Statistics Department, University of California, Berkley, CA 94720, January 2001

¹⁸ Leo Breiman & Adele Cutler, *Random Forests*, [Website] http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm
Theoretical underpinnings of page are laid out in paper “Random Forests” (Breiman, 2001)

gives internal estimates that are helpful in understanding classification accuracy and how to improve it" (Breiman, 2001). For details about this approach, please consult: RANDOM FORESTS, Leo Breiman, 2001.

3.3.3 Balanced Random Forest

As was discussed in the chapter about imbalanced data, Chen et. al. (2005)¹⁹ describe the process of doing modifying bootstrap samples and come up with a Balanced Random Forest (BRF) algorithm. The distinguishing factor in their algorithm works as follows: *"For each iteration in random forest, draw a bootstrap sample from the minority class. Randomly draw the same number of cases, with replacement, from the majority class."* (Chen et. al, 2005). Assume K is the number of minority cases in a training set N . Instead of sampling N cases at random from the training set (step 1 in the Random Forest scheme), sample K cases, with replacement, from the minority class. Continue by sampling K cases, with replacement, from the majority class. This results in a balanced training set to build a tree in the forest. The Balanced Random Forest scheme

Lets assume the number of cases in the training set is N and there are M input variables, also there are **K cases of the minority class**. Now for every new tree:

1. With replacement, **sample $k \leq K$ minority cases** at random from the training set. Also **sample $k \leq K$ majority cases** at random from the training set. Together, these samples will be the training set for the single tree (a balanced training set).
2. Specify a number $m \ll M$ such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. This value m is held constant during the forest growing.
3. There is no pruning involved. Each tree is grown to the largest extent possible.

Table 3.4: Tree growing scheme for a tree in a Balanced Random Forest

In their study, Chen et. al (2005) establish that BRF is computationally more efficient with large imbalanced data than its counterpart; the Weighted Random Forest (WRF). The reason behind this, is that BRF only uses a small portion of the training set to grow, while WRF needs to use the entire training set. It is also concluded that WRF is possibly more vulnerable to noise than BRF.

3.3.4 Advantages

According to Breiman (2001) There are many advantages to using Random Forest. First of all it is unexcelled in accuracy among many current algorithms. It is relatively robust to outliers and noise. It runs computationally efficiently on large databases and is faster than bagging or boosting. Breiman & Cutler expand this list by stating that Random Forests can handle thousands of input variables without variable deletion and that they do not overfit. One can run as many trees as desired while the forest is generated very fast. Another advantage is, just like with normal decision trees, a Random Forest can handle combinations of numerical and categorical data. Also, scaling is not required because Random Forest training is invariant to all combinations of monotonic transformations of predictors. Last but not least, a BRF solves the problem of dealing with imbalanced data.

There is much more to write about Random Forests. For example, measures for variable importance, gini importance, interactions and proximities. However, these are beyond the scope of this paper. For more information, again, consult: RANDOM FORESTS, Leo Breiman, 2001. This chapter has tried to explain the theoretical basics of Random Forests. The coming chapter will describe why this makes Random Forests particularly interesting for the Kaggle competition.

¹⁹ Chao Chen, Andy Liaw, Leo Breiman (2004), *Using Random Forest to Learn Imbalanced Data*.

4. Method

Various techniques to solve data-related issues, like the presence of text and the data being imbalanced, have been presented in previous chapters. Therefore, this chapter tries not only to justify the techniques selected, but also gives the main reasons for selecting Random Forest as the final model. The chapter can be viewed as the summation of previous theoretical chapters resulting in a final model and its parameters. Also, the features generated are discussed.

As was stated earlier, an algorithm for this challenge needs to fulfill a few criteria. First of all, it needs to be able to handle mixed data and handle large databases with many input variables. More importantly however, is the desire for an algorithm that works fast and is computationally efficient. Participating in a competition for just one month, requires many runs of the algorithm in a short time. Random Forest is clearly not the only model that fulfils these conditions. However, through literature research and by simply applying different models to the data, other candidates were eliminated. Support Vector Machines (SVM) and Neural Networks were some of the eliminated candidates. Because Random Forests are fast, scaling is unnecessary, are insensitive to correlating variables, there are many methods to help Random Forests handle imbalanced data, and there is only little parameter tuning required, it is an ideal algorithm for the Kaggle competition.

4.1 Balanced Random Forest

The table below shows the settings for the Balanced Random Forest. These are actually the only 3 variables that makes sense (see chapter on Random Forests). Number of trees is optimized through trial and error since there simply is no optimization for the number of bootstrap replicates. Taking $k = K$ makes sense knowing there is small number of minority cases it is preferred to take this sample size to as large as possible. The number of variables selected at random from the input variables is often calculated by: $m = \sqrt{M}$. Optimizing this variable turns out to be difficult. It easily disturbs the relation between tree strength and correlation (see chapter on Random Forests).

m = Number of variables selected at random from the input variables (M) $m \ll M$	Number of trees	Number of sampled minority cases k
$m = \sqrt{M}$	5001	$k = K$

Table 4.1: Taking $m = \sqrt{M}$ is a widely used size for m . The number of trees is often set to an uneven number to break ties. The number of sampled minority cases $k=K$ uses as much minority cases as possible (with replacement).

4.2 Feature generation

Like with any machine learning task, feature generation is the difference between failure and success. For this task the feature generation can be divided into three categories: simple adjustments to numerical or categorical features, text mining features and features based on the overlapping features. For a list of all the features used in the model, please consult the Appendix.

4.2.1 Numerical and categorical

Many numerical and categorical features were generated. However, the majority of these features did not prove to be useful. Some features were even based on external data about education quality per state. In the end, only two features turned out to improve the model. These features were calculated using two existing features. They are listed in the table below:

Feature:	Formula:
total_price_per_student_excl	$\text{total_price_excluding_optional_support} / \text{students_reached}$
total_price_per_student_incl	$\text{total_price_including_optional_support} / \text{students_reached}$

Table 4.2: Two generated formulas from existing features in the data set.

4.2.2 Text

In the theoretical chapters, some text mining techniques were explained. For the Kaggle competition however, with the few text attributes present, the trade-off between model improvement and computation time resulted in the decision not to use TF*IDF. The data pre-processing techniques like stemming and summarization techniques like lexical diversity did prove to be useful. The table below shows the generated features using text mining techniques:

After removing punctuation and removing numbers:	After removing punctuation, removing numbers and stemming the texts:
length_title words_per_title wsize_per_title	num_uniq_words_title lex_div_title
length_short_description words_per_short_description wsize_per_short_description	num_uniq_words_short_description lex_div_short_description
length_need_statement words_per_need_statement wsize_per_need_statement	num_uniq_words_need_statement lex_div_need_statement
length_essay words_per_essay wsize_per_essay	num_uniq_words_essay lex_div_essay

Table 4.3: Generated numerical features using text mining techniques

4.2.3 Overlap approach

In the data exploration chapters it was already concluded that for features like school – and teacher id there is overlap. The test set also contains new teacher id's of which there is no knowledge. However, features like “teacher_funding_request_frequency” (how many times did the teacher request funds) and what has been their success rate up until that request, are also interesting. The frequency can be informative because being successful in requesting funds might be a learning process. A second request could be more successful than a first. The success rate shows something similar. How successful is a particular teacher (or school). These features can help predicting an exciting project.

The frequency feature can be inserted quite easily. The training- and test set can be adjusted in a way that frequencies are included. The success rate on the other hand, is a little more difficult. The table below shows the result of what has been explained so far. The training set is adjusted to a training set that includes frequency and success rate.

Trainingset		Trainingset			
Teacher_ID	is_Exiting	Teacher_ID	Freq	Succes rate	is_Exiting
A	1	A	1	MISSING	1
A	1	A	2	1	1
B	1	B	1	MISSING	1
B	0	B	2	1	0
B	0	B	3	0.5	0
C	1	C	1	MISSING	1

Table 4.2: Including teacher_frequency and teacher_success_rate in the training set.

The missing values in the success rate column can be replaced with the mean success rate on the first request. In the test set however, a new teacher id may occur more than once (a teachers first, second and third request can occur). Therefore, for the test set success rate, at frequency N, the success rate is replaced by the mean success rate at request N.

This results in the following new features:

Features
Teacher_Frequency
Teacher_Success_Rate
School_Frequency
School_Success_Rate
State_Frequency
State_Success_Rate
County_Frequency
County_Success_Rate

Table 4.3: *Overlap features*

4.3 Variable inclusion

Variable inclusion is an important aspect. For this particular assignment, a typical trial-and-error method is advisable. This is due to the many different variables and time being scarce. This trial-and-error method starts with a basic solution with a few variables (selected by intuition). From there on new variables are included and the leaderboard score is evaluated. If the attribute(s) improve the score, they should be included. Optimizing leaderboard scores like this, is a good alternative to advanced variable importance measures which consume more time.

5. Results and Discussion

Over the duration of this project many different models were attempted. The previous chapters have already shown why a BRF is a suitable model for this particular problem. However, in the early stages of the project different possibilities were also investigated. The graph below shows the increasing leaderboard score on Kaggle.com with the corresponding modeling attempts.

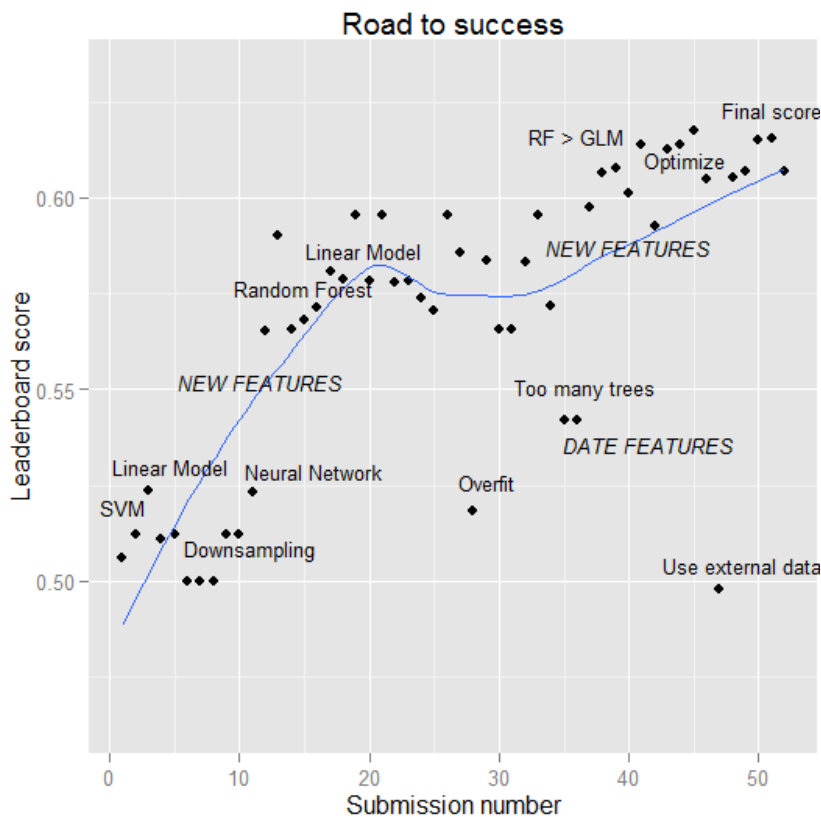


Figure 5.1: *Increasing leaderboard score on Kaggle.com with the corresponding modeling attempts. The final leaderboard score: 0.61545*

A Random Forest was not the only possible model. Neural Networks (NN) seemed an interesting option as well. However, a Neural Network is not as computationally efficient and fast as a Random

Forest. Also, finding optimal parameters is an important aspect of a Neural Network. Random Forests are more user-friendly (faster, less parameter tuning). An SVM also proved to be too slow for the task at hand.

In the early stages, the down sampling (see graph) approach was different. Before applying any model, entries were deleted until the data was balanced (50/50). Doing so, means throwing away valuable data. However, when the first BRF and the text features were included around submission 10, a giant increase in score was achieved.

At that point (submission 20) a linear model also proved to be fast and with promising results. However, when more and more features were introduced the Balanced Random Forest worked much better. The “overlap approach” eventually eliminated any remaining doubt and showed that BRF was the best algorithm for the job. Furthermore, it turned out that date features performed poorly and an attempt to include external data about education quality and economic measures per state performed poorly. The final result of this competition is a 47th position on the Kaggle leaderboard:

44	—	mmn	0.61603	54	Tue, 15 Jul 2014 10:33:40 (-0.9h)
45	↑3	shss	0.61576	52	Tue, 15 Jul 2014 08:52:19 (-3.2h)
46	↓1	shzu-IntCMP-ML01	0.61565	109	Tue, 15 Jul 2014 20:53:03 (-13.7h)
47	↓8	AntonK	0.61545	52	Sun, 29 Jun 2014 09:02:10 (-8.8h)
48	↑79	yoyow110w	0.61410	24	Mon, 14 Jul 2014 15:20:48 (-0h)

Figure 5.2: The final leaderboard position (47th).

Important variables to achieve this result turned out to be “focus subject” related, price related and location related. The generated text mining features and the frequency/succes_rate features also improved the model drastically. As was stated earlier, a trial-and-error method was used for variable inclusion. Optimizing leaderboard scores like this, is a good alternative to advanced variable importance measures which consume more time. However, doing so does not allow a thorough analysis about the importance of the variable within the model. There are measures to do so but that is beyond the scope of this paper. Nevertheless, for next time, it is an interesting option to change the feature inclusion technique. Instead of trial-and-error, one could use different variable importance measures for Random Forests (like Mean Decrease Gini). This is definitely something that could have been researched more extensively. This could result in a more convincing chapter on variable importance.

5.1 Validation

When it comes to validation and evaluation of the models, Kaggle provided two test sets: a temporary test set and a final test set. The final leaderboard ranking was determined using the so-called “final test set”. However, throughout the competition, any model could be validated using the temporary test set (actually a validation set). This test set consisted of the final test set’s first 40% of entries. Because of this, a cross validation method was not really necessary. I simply submitted the classifications and both score and ranking (using test set) were computed.

On the final day of the competition, the final scores for all submissions were revealed. In other words, on the closing day of the competition, for each validation submission I made, the performance using the final test set was revealed as well. Only two preselected submissions would be used for the final ranking.

The graph below shows how performance of the model using the test sets related to the nth submission. Over the different submissions, both sets follow the same pattern. However, the final test score is always slightly lower.

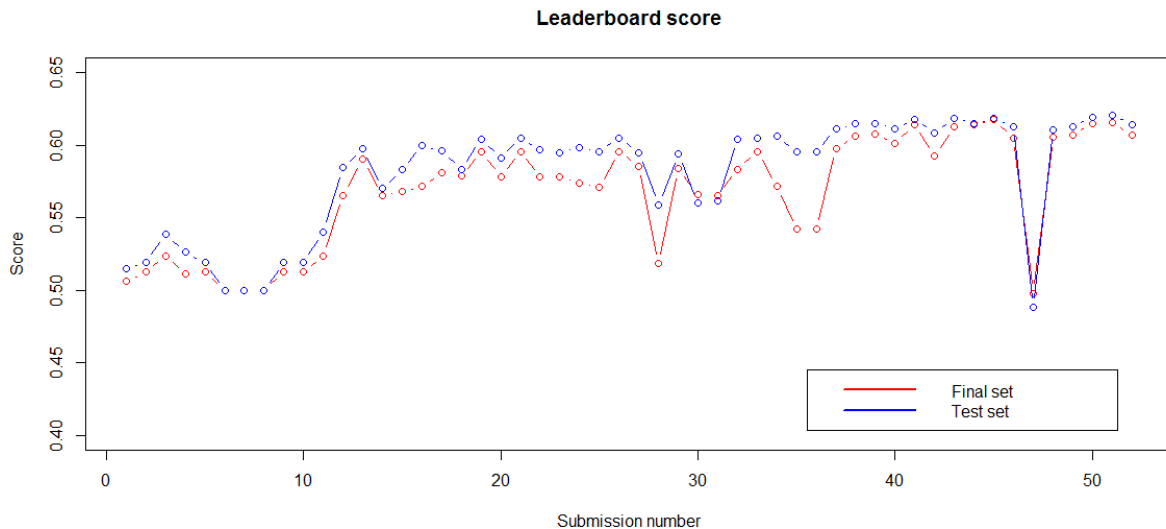


Figure 5.3: Submission scores in chronological order based on final- and test set.

At submission 28 and between 34 and 36 there are surprising differences. The absolute deviations are much higher here than on average. The table below illustrates this:

#	Scores Temp	Final	Absolute deviation
28	0,5590	0,5182	0,0408
34	0,6059	0,5716	0,0343
35	0,5954	0,5420	0,0534
36	0,5954	0,5420	0,0534
			MAD=0,0126

Table 5.1: Maximum differences between final- and test set. These submission had in common that they all included date attributes. Therefore, one could say that due to differences between the two sets we can confirm that date related attributes are indeed difficult to use.

The different submissions exhibiting these large deviations (compared to the MAD of all the submissions) are completely different. Submission 34 for example used a GLM, while the others used Random Forests with completely different parameters and attributes. However, they all had one thing in common. They all included date related attributes. None of the other submissions contained these since they proved to be very hard to utilize. An offline validation method I programmed did show improvement when including date related features. However, Kaggle's online test set showed worse and worse results when including more date related features. This confirmed the idea of doing all the validation online with the provided temporary test set. Many other competitors also struggled with date related attributes. Consulting the competition's forum reveals that these competitors used the fact that throughout a year the number of exciting projects slowly decreases. In other words, in the beginning of an academic year people are more likely to donate to school projects. This is often referred to as decay. Assuming that teachers who discover DonorChoose.org are likely to make new donation requests within the same year, one could say this decay is also (very) mildly included in the BRF. This is due to the inclusion of teacher frequency and success rate. Nevertheless, applying a linear decay could have improved the final score and allowed inclusion of date related attributes.

Obviously, within a competition like this, many decisions have to be made. Examples of these are: How advanced will the text mining models be? How advanced should the missing value techniques be? What is the trade-off between a computationally efficient algorithm and an algorithm that could maximize the score? Etc. However, it is impossible to substantiate every decision and I think that's what makes the field of machine learning interesting. Sometimes it is just a matter of trial-and-error.

7. Conclusion

Predicting funding requests that are exciting can be challenging. In this research the problem was analyzed and based on typical properties of this problem models and techniques were selected. There were a number of properties that were taken into account when solving the problem. The two most important properties were the imbalanced nature of the data and the presence of text attributes. Techniques to deal with these data characters were researched and described in this paper. A proven technique to deal with imbalanced data is down sampling. Many Machine Learning algorithms take bootstrap samples from the training set. These samples can be modified, resulting in balanced samples without loss of information (as is the case with traditional down sampling). Therefore, this method was implemented in the final model.

The second complicating factor was the presence of text attributes. Again, after thorough research, text mining techniques were used to summarize text using new numerical attributes like lexical diversity, average word length, text length and number of words.

The fact that the data was mixed (numerical and categorical) also required attention when choosing the model. Another important requirement for an algorithm was that it had to be fast. I only participated in the competition for one month. Algorithms with extreme running times would cause major problems.

Taking the above into account, Balanced Random Forest (BRF) is an example of a suitable algorithm. There are many advantages to using a BRF. For example, BRF does not require scaling, is fast, imbalanced data or mixed data is no problem, is robust to outliers and noise and requires little parameter tuning.

When it comes to feature generation, some new features were created using the text summarization techniques. Other features were created by adjusting existing features. However, one of the most important new features was created using the overlapping features (school id and teacher id). This method created two new features: success rate and frequency.

By applying these techniques the goal (top 50) for this research was achieved with a ranking of 47/472. Therefore, BRF is an incredibly effective algorithm for large, imbalanced and mixed data problems with some text attributes. In conclusion, sufficient data understanding in combination with theoretical understanding of approaches to tackle data related issues does not require advanced Machine Learning models. A slightly adjusted Random Forest model showed impressive (suboptimal) results. Going through the forums of Kaggle.com shows that many top 50 competitors used more advanced techniques.

APPENDIX 1

1.1 Extra Information and Research question as posted on Kaggle.com

Extra Information

This Research Paper project started on June 1st (2014) and a final submission to Kaggle was made on the 29th of June. The Kaggle contest however, was active from May 16th until July 16th. For the competition, an HP Ultrabook with 16 GB RAM was used. Computations were carried out in R, a free software environment for statistical computing and graphics.

Throughout July and September parts of this paper were reviewed and rewritten resulting in a final paper which was finished at the end of September 2014.

Predict funding requests that deserve an A+



DonorsChoose.org is an online charity that makes it easy to help students in need through school donations. At any time, thousands of teachers in K-12 schools propose projects requesting materials to enhance the education of their students. When a project reaches its funding goal, they ship the materials to the school.

The 2014 KDD Cup asks participants to help DonorsChoose.org identify projects that are exceptionally exciting to the business, at the time of posting. While all projects on the site fulfill some kind of need, certain projects have a quality above and beyond what is typical. By identifying and recommending such projects early, they will improve funding outcomes, better the user experience, and help more students receive the materials they need to learn.

Successful predictions may require a broad range of analytical skills, from natural language processing on the need statements to data mining and classical supervised learning on the descriptive factors around each project.

1.2 Advantages of competing in a Kaggle competition

Competing in a Kaggle competition guarantees scientific or social relevance. This is confirmed by the amount of money (\$2000) that is awarded to the most successful solutions to this particular problem. Almost 500 teams competed in this challenge. Therefore, the final ranking in the competition serves as an indication of the quality of the research.

Thirdly, a study²⁰ on the importance of and time spent on different modeling steps states that approximately 20% of time is spent on data collection. Since the Research Paper project is recommended to be carried out within a month, the already collected and slightly cleaned-up data of a Kaggle competition is definitely an advantage. The clearly formulated problem is also an important reason for choosing a Kaggle competition. This enables a focus on solving the problem instead of formulating it and leaves more opportunity for theoretical background research to substantiate any modeling decisions.

Some notes:

The predefined research period was a single month (June 2014). The Kaggle competition however, was open for two months. This means I withheld myself from new submissions after June 30. Therefore the final ranking of this model could have been even better. My top 10% ranking proves the significance of this paper and suggests that when attempting to remove a number of the practical constraints the results could have been even better. In other words: even though limited in a number of ways, my top 50 goal in the Kaggle competition was achieved without extensive Data Mining/Machine Learning experience.

²⁰ M. Arthur Munson (2011), A study on the importance of and time spent on different modeling steps, ACM SIGKDD Explorations newsletter, v.13 n.2, December 2011 <http://dl.acm.org/citation.cfm?id=2207243.2207253>

Appendix 2

Appendix 2.1-2.4 are directly taken from Kaggle:

<http://www.kaggle.com/c/kdd-cup-2014-predicting-excitement-at-donors-choose/data>

2.1 Data

The data is provided in a relational format and split by dates. Any project posted prior to 2014-01-01 is in the training set (along with its funding outcomes). Any project posted after is in the test set. Some projects in the test set may still be live and are ignored in the scoring. We do not disclose which projects are still live to avoid leakage regarding the funding status.

2.2 File descriptions

- **donations.csv** - contains information about the donations to each project. This is only provided for projects in the training set.
- **essays.csv** - contains project text posted by the teachers. This is provided for both the training and test set.
- **projects.csv** - contains information about each project. This is provided for both the training and test set.
- **resources.csv** - contains information about the resources requested for each project. This is provided for both the training and test set.
- **outcomes.csv** - contains information about the outcomes of projects in the training set.
- **sampleSubmission.csv** - contains the project ids of the test set and shows the submission format for the competition.

2.3 "Exciting" Projects

Exciting projects meet a number of requirements specified by DonorsChoose.org. Note that the term "exciting" is meant as a business construct and does not imply that non-exciting projects are not compelling to teachers/students/donors! To be exciting, a project must meet **all** of the following five criteria. The name in parentheses indicates the field containing each feature in the data set.

- was fully funded (fully_funded)
- had at least one teacher-acquired donor (at_least_1_teacher_referred_donor)
- has a higher than average percentage of donors leaving an original message (great_chat)
- has at least one "green" donation (at_least_1_green_donation)

- has one or more of:
 - donations from three or more non teacher-acquired donors
(three_or_more_non_teacher_referred_donors)
 - one non teacher-acquired donor gave more than \$100
(one_non_teacher_referred_donor_giving_100_plus)
 - the project received a donation from a "thoughtful donor"
(donation_from_thoughtful_donor)

You will find this information summarized in outcomes.csv, including the boolean value for is_exciting.

2.4 Data fields

Below is a brief explanation of the provided data fields. Descriptions of self-explanatory names are omitted.

outcomes.csv

is_exciting - ground truth of whether a project is exciting from business perspective

at_least_1_teacher_referred_donor - teacher referred = donor donated because teacher shared a link or publicized their page

fully_funded - project was successfully completed

at_least_1_green_donation - a green donation is a donation made with credit card, PayPal, Amazon or check

great_chat - project has a comment thread with greater than average unique comments

three_or_more_non_teacher_referred_donors - non-teacher referred is a donor that landed on the site by means other than a teacher referral link/page

one_non_teacher_referred_donor_giving_100_plus - see above

donation_from_thoughtful_donor - a curated list of ~15 donors that are power donors and picky choosers (we trust them selecting great projects)

great_messages_proportion - how great_chat is calculated. proportion of comments on the project page that are unique. If > avg (currently 62%) then great_chat = True

teacher_referred_count - number of donors that were teacher referred (see above)

non_teacher_referred_count - number of donors that were non-teacher referred (see above)

projects.csv

projectid - project's unique identifier

teacher_acctid - teacher's unique identifier (teacher that created a project)

schoolid - school's unique identifier (school where teacher works)

school_ncesid - public National Center for Ed Statistics id

school_latitude

school_longitude

school_city

school_state

school_zip

school_metro

school_district

school_county

school_charter - whether a public charter school or not (no private schools in the dataset)

school_magnet - whether a public magnet school or not

school_year_round - whether a public year round school or not

school_nlns - whether a public nlns school or not

school_kipp - whether a public kipp school or not

school_charter_ready_promise - whether a public ready promise school or not

teacher_prefix - teacher's gender

teacher_teach_for_america - Teach for America or not

teacher_ny_teaching_fellow - New York teaching fellow or not

primary_focus_subject - main subject for which project materials are intended

primary_focus_area - main subject area for which project materials are intended

secondary_focus_subject - secondary subject

secondary_focus_area - secondary subject area

resource_type - main type of resources requested by a project

poverty_level - school's poverty level.

highest: 65%+ free of reduced lunch

high: 40-64%

moderate: 10-39%

low: 0-9%

grade_level - grade level for which project materials are intended

fulfillment_labor_materials - cost of fulfillment

total_price_excluding_optional_support - project cost excluding optional tip that donors give to DonorsChoose.org while funding a project

total_price_including_optional_support - see above

students_reached - number of students impacted by a project (if funded)

eligible_double_your_impact_match - project was eligible for a 50% off offer by a corporate partner (logo appears on a project, like Starbucks or Disney)

eligible_almost_home_match - project was eligible for a \$100 boost offer by a corporate partner

date_posted - data a project went live on the site

donations.csv

donationid - unique donation identifier

projectid - unique project identifier (project that received the donation)

donor_acctid - unique donor identifier (donor that made a donation)

donor_city

donor_state

donor_zip

is_teacher_acct - donor is also a teacher

donation_timestamp

donation_to_project - amount to project, excluding optional support (tip)

donation_optional_support - amount of optional support

donation_total - donated amount

dollar_amount - donated amount in US dollars

donation_included_optional_support - whether optional support (tip) was included for DonorsChoose.org

payment_method - what card/payment option was used

payment_included_acct_credit - whether a portion of a donation used account credits redemption

payment_included_campaign_gift_card - whether a portion of a donation included corporate sponsored giftcard

payment_included_web_purchased_gift_card - whether a portion of a donation included citizen purchased giftcard (ex: friend buy a giftcard for you)

payment_was_promo_matched - whether a donation was matched 1-1 with corporate funds

via_giving_page - donation given via a giving / campaign page (example: Mustaches for Kids)

for_honoree - donation made for an honoree

donation_message - donation comment/message. Used to calculate great_chat

essays.csv

projectid - unique project identifier

teacher_acctid - teacher id that created a project

title - title of the project

short_description - description of a project

need_statement - need statement of a project

essay - complete project essay

resources.csv

resourceid - unique resource id

projectid - project id that requested resources for a classroom

vendorid - vendor id that supplies resources to a project

vendor_name

project_resource_type - type of resource

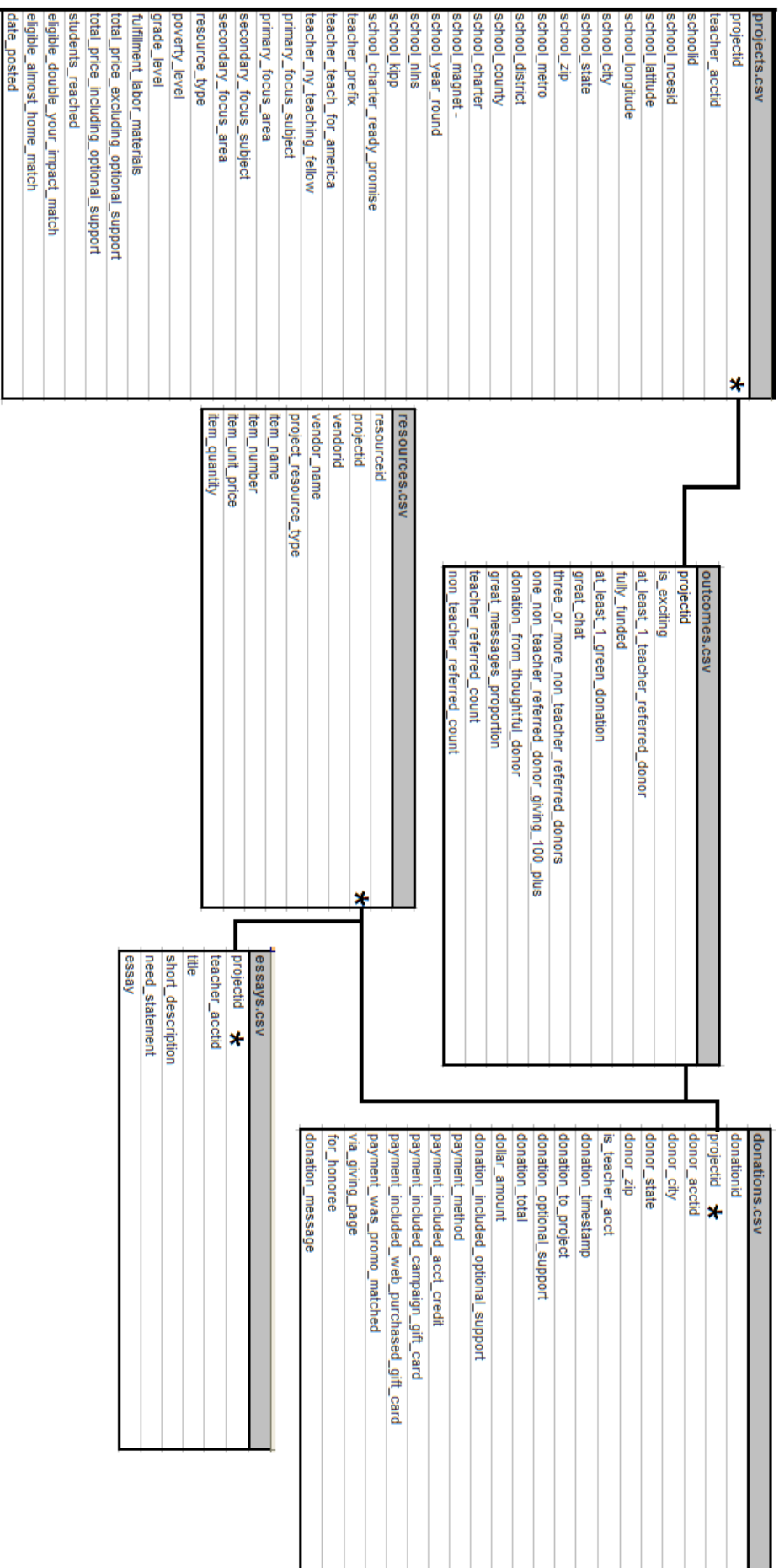
item_name - resource name (ex: ipad 32 GB)

item_number - resource item identifier

item_unit_price - unit price of the resource

item_quantity - number of a specific item requested by a teacher

Overview of the data provided



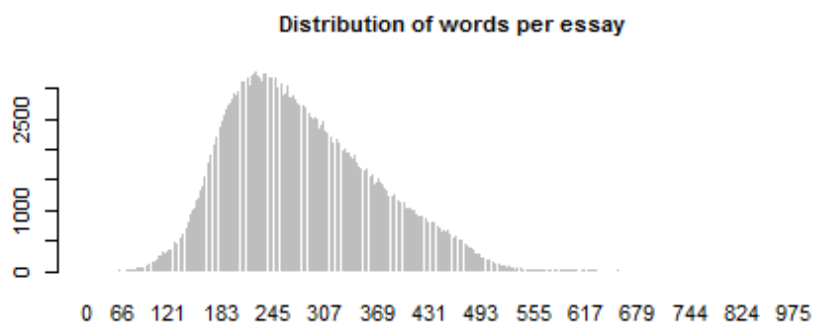
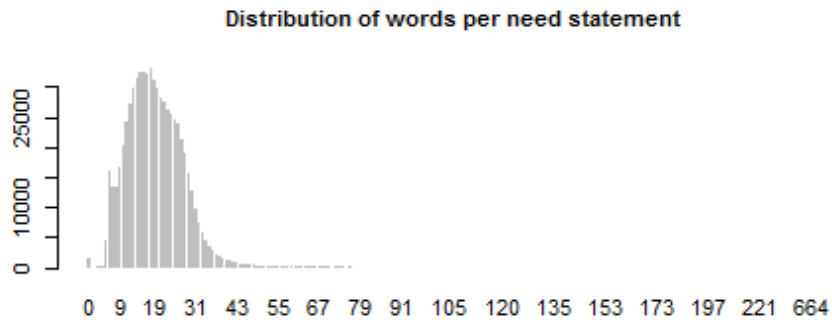
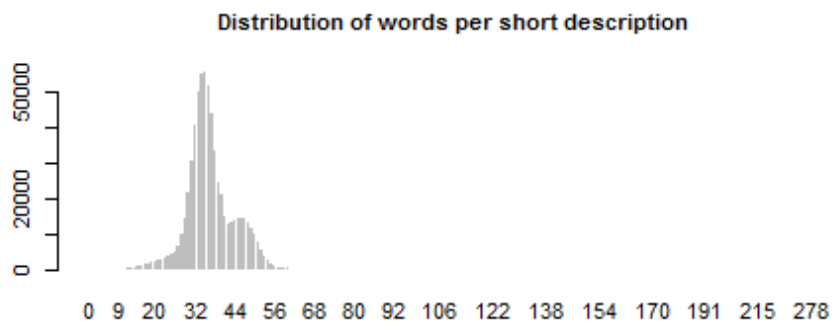
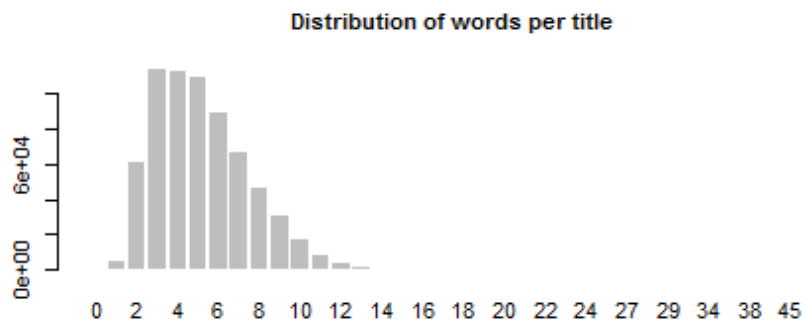
Appendix 3

3.1 Kaggle leaderboard

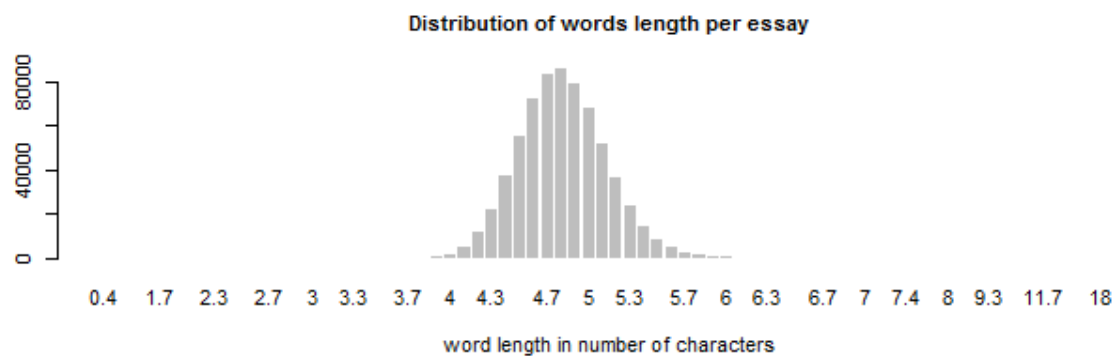
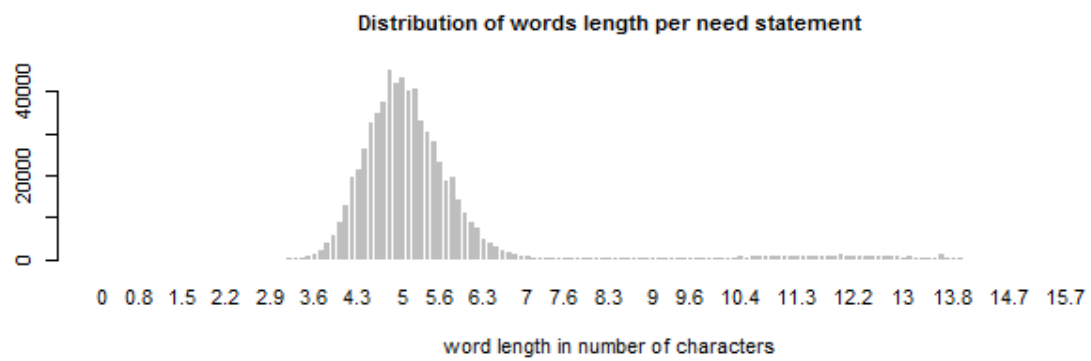
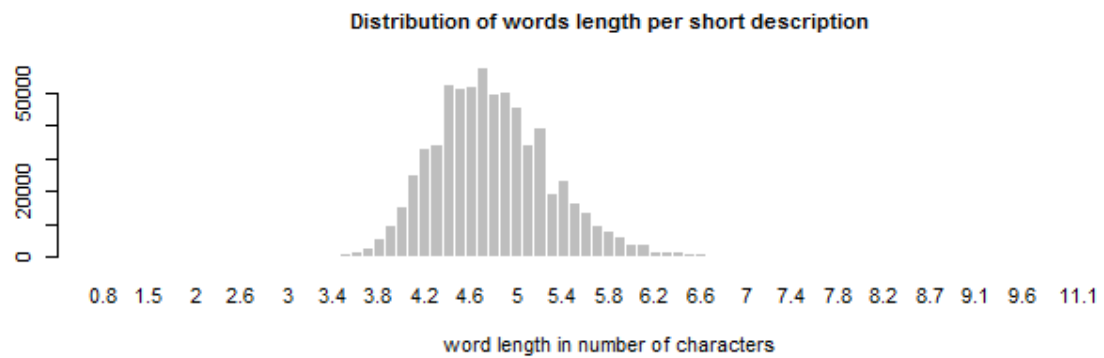
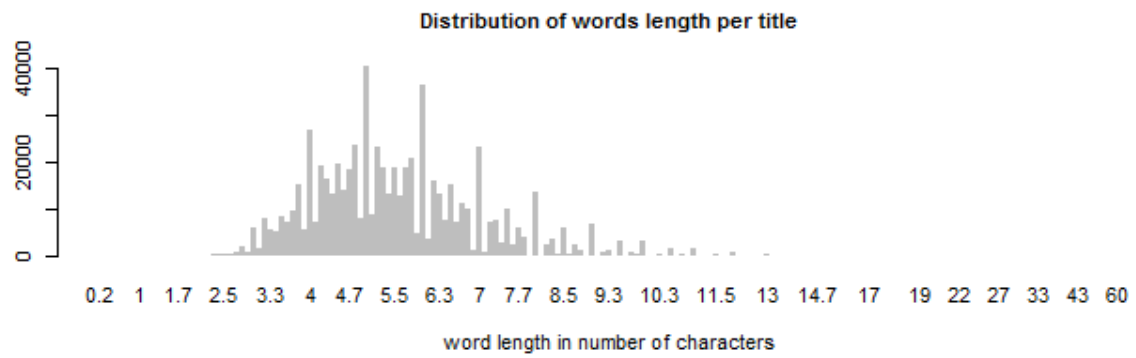
#	Δ1w	Team Name <small>† in the money</small>	Score <small>📊</small>	Entries	Last Submission UTC (Best - Last Submission)
39	↓10	FEG <small>🏆</small>	0.61935	153	Tue, 15 Jul 2014 15:29:51 (-25.9h)
40	↑149	Marcin Mucha	0.61888	25	Fri, 11 Jul 2014 13:32:25
41	↓4	Johan Edvinsson	0.61763	6	Sun, 15 Jun 2014 16:56:25 (-41.8h)
42	↓32	BonneNuit	0.61739	38	Tue, 15 Jul 2014 07:17:43 (-1.5h)
43	↑272	RH	0.61606	10	Mon, 14 Jul 2014 22:18:30 (-0.2h)
44	—	mmn	0.61603	54	Tue, 15 Jul 2014 10:33:40 (-0.9h)
45	↑3	shss	0.61576	52	Tue, 15 Jul 2014 08:52:19 (-3.2h)
46	↓1	shzu-IntCMP-ML01	0.61565	109	Tue, 15 Jul 2014 20:53:03 (-13.7h)
47	↓8	AntonK	0.61545	52	Sun, 29 Jun 2014 09:02:10 (-8.8h)
48	↑79	yoyow110w	0.61410	24	Mon, 14 Jul 2014 15:20:48 (-0h)
49	↓9	BookEx.org	0.61313	7	Tue, 10 Jun 2014 10:30:17 (-22.4h)
50	↓4	Catalyst@UOM <small>🏆</small>	0.61198	82	Tue, 15 Jul 2014 15:42:49
51	↓9	Skeeter	0.61125	9	Sun, 25 May 2014 22:58:12
52	—	Mockingjay@UOM <small>🏆</small>	0.61016	33	Tue, 15 Jul 2014 19:36:51 (-2.1d)
53	↓3	502B <small>🏆</small>	0.60886	54	Sun, 13 Jul 2014 11:24:38 (-4.3d)
54	↑46	yakku@uom <small>🏆</small>	0.60862	71	Tue, 15 Jul 2014 10:36:58 (-0.1h)
55	↑15	sasan	0.60818	74	Sat, 12 Jul 2014 14:28:02 (-0.3h)
56	↓9	Dmitry Efimov	0.60812	18	Sun, 22 Jun 2014 19:46:11 (-0.1h)
57	↑141	TheAnalyticProphet	0.60626	22	Tue, 15 Jul 2014 08:34:54 (-32.7h)
58	↑176	Atuts Guys	0.60564	5	Tue, 15 Jul 2014 19:09:51
59	↓16	Jeong-Yoon, Feng & Hang <small>🏆</small>	0.60527	120	Tue, 15 Jul 2014 17:35:32 (-3.3h)



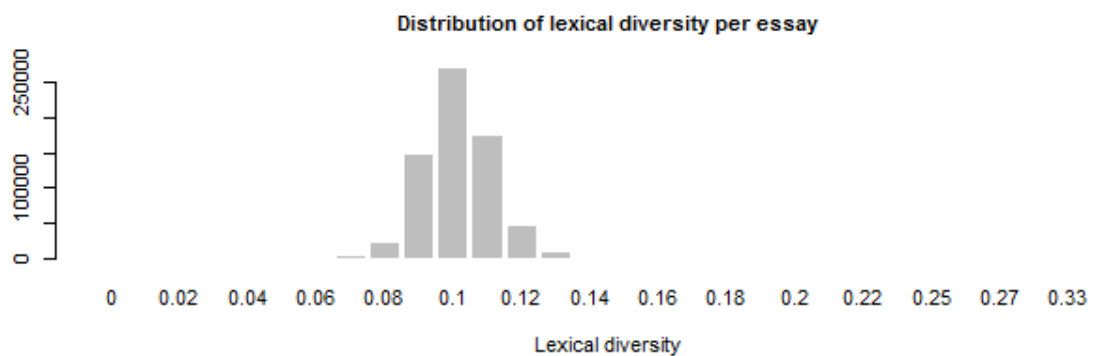
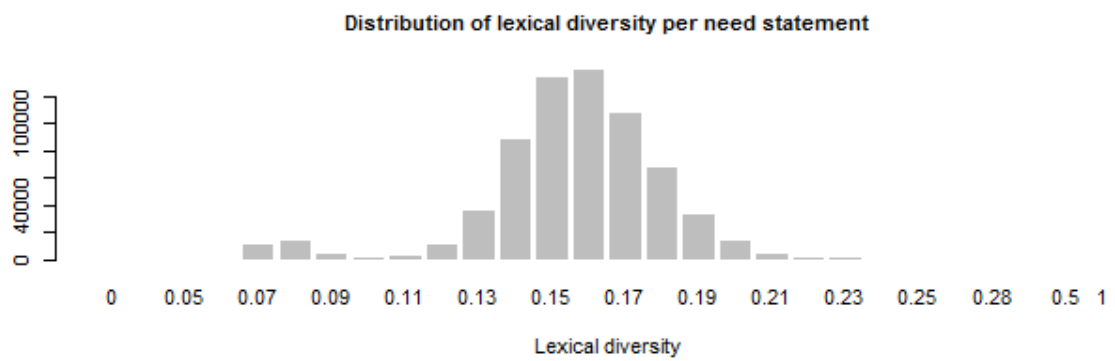
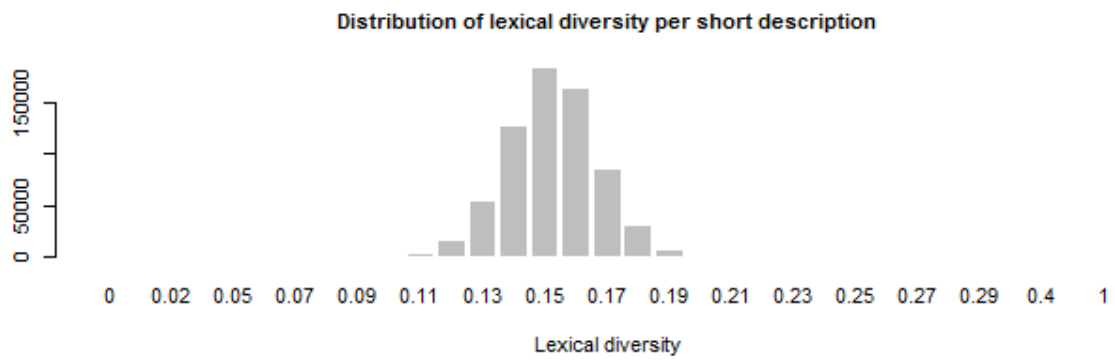
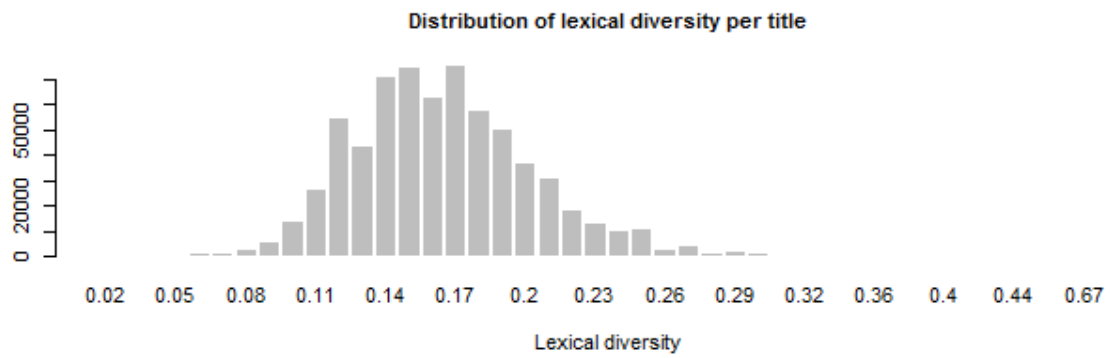
3.2 Text mining graphs



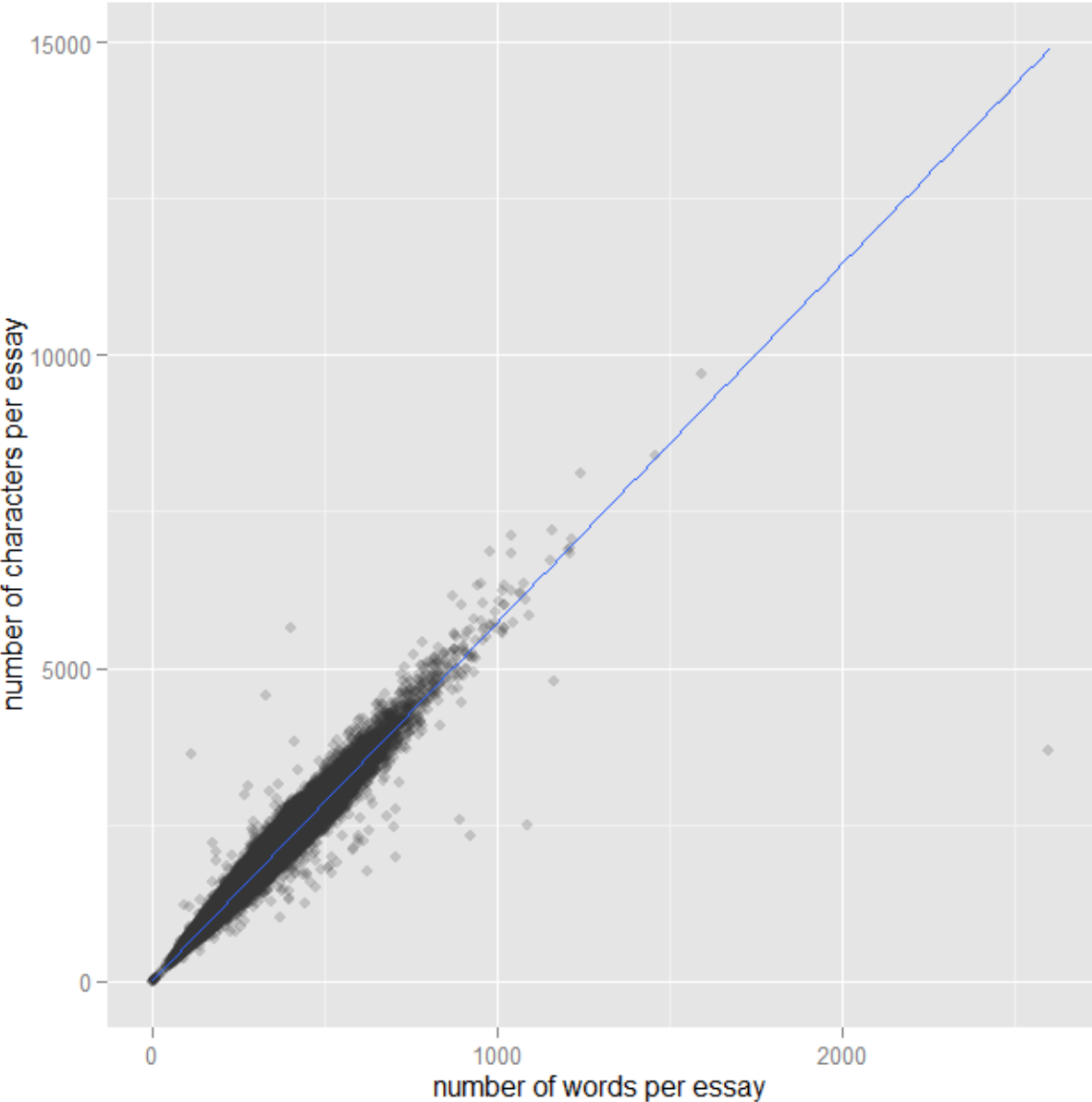
3.2 Text mining graphs



3.2 Text mining graphs



3.2 Text mining graphs



3.2 Text mining graphs

