UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Krister Jaanhold

Automated Analysis of Customer Contacts – a Fintech Based Case Study

Master's Thesis (30 ECTS)

Supervisors: Sven Laur, DSc
Taavi Tamkivi, MSc

Tartu 2018

Declaration of authorship

I hereby declare that given thesis and the work presented in it is my own. I confirm that:

- This work was done wholly or mainly while in candidature for a degree at the University of Tartu.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

…………………………………..

Krister Jaanhold

Automated Analysis of Customer Contacts – a Fintech Based Case Study

Abstract:

The rapid development of information technologies has brought along abnormal amounts of data being generated on a daily basis and the need to automatically analyse it to gain a competitive advantage. Traditional data mining techniques have been efficiently applied in a variety of commercial applications, yet they are only applicable on structured data. However, an overwhelming amount of existing data is in an unstructured (e.g. textual) form, hence it is crucial for companies to build solutions to automatically extract useful information from it. Given master's thesis is with a practical nature and its purpose was to implement an automated text analysis model using data from TransferWise Ltd. that can be used to efficiently prioritise and measure incoming customer contacts. To achieve this, the author conducted numerous experiments via employing classical as well as novel natural language processing techniques. Apropos, employing novel methods did not ensure a noticeably better outcome. The established model is important for both the company as well as its customers since it can be used to prioritise incoming contacts based on their complexity or urgency. This ensures a convenient customer experience and is likely to accelerate growth by making operational procedures more efficient. Besides its practical value, given thesis also provides an extensive comparison of numerous natural language processing techniques, their suitability and opportunities.

Keywords: text analysis, natural language processing, business intelligence, financial technologies.

CERCS: P170 – computer science, numerical analysis, systems, control.

Automatiseeritud kliendikontaktide analüüs finantstehnoloogia sektori näitel

Lühikokkuvõte:

Seoses infotehnoloogia arenguga tekib igapäevaselt enneolematu kogus andmeid, mille automaatne analüüsimine konkurentsieelise saavutamiseks on otsustava tähtsusega. Traditsioonilised andmekaeve meetodid on leidnud laialdaselt ärilisi rakendusi, kuid ei ole sobivad struktureerimata (näiteks tekstiliste) andmete puhul. Seevastu on valdav osa andmetest just struktureerimata kujul, mistõttu on iseäranis oluline luua lahendusi neist olulise teabe eraldamiseks. Käesolev magistritöö on praktilise loomuga ning selle eesmärk oli luua automatiseeritud tekstianalüüsi mudel, mida saab kasutada sissetulevate kliendipäringute efektiivseks prioriseerimiseks ning mõõtmiseks kasutades TransferWise Ltd. andmeid. Tulenevalt püstitatud eesmärgist teostas autor arvukalt eksperimente kasutades nii klassikalisi kui ka uudseid loomuliku keele töötluse meetodeid. Seejuures ei taganud antud ülesande puhul uudsed tehnoloogiad märgatavat paremust klassikaliste meetodite ees. Töö tulemusena valminud mudel on oluline nii ettevõttele kui ka selle klientidele – mudel võimaldab prioriseerida sissetulevaid päringuid vastavalt nende keerukusele ning pakilisusele, mis parandab kliendikogemust ning soodustab ettevõtte kasvu muutes operatsioonilisi protsesse efektiivsemaks. Peale praktilise väärtuse pakub käesolev töö ka ulatuslikku ülevaadet erinevatest loomuliku keele töötluse meetoditest, nende sobivusest ning nendega kaasnevatest võimalustest.

Võtmesõnad: tekstianalüüs, loomuliku keele töötlus, ärianalüüs, finantstehnoloogiad.

CERCS: P170 – arvutiteadus, arvutusmeetodid, süsteemid, juhtimine.

Contents

1 Introduction

The evolution of information technologies has led to unprecedented amounts of data being generated on a daily basis. To gain an advantage over its competitors, companies strive to make intelligent use of their data – automatically discovering patterns in large, structured datasets has been employed in a variety of commercial applications (Zhang & Zhou, 2004: 516-519). However, the majority of available data is in an unstructured, textual form (Gupta *et al.,* 2009: 72) which makes building solutions to automatically extract useful information from it a high priority. This process is referred to as text mining and it is considered to be a powerful extension to traditional decision support systems (Gao *et al.,* 2007: 110). An abundant source of information for businesses is customer contacts – there are thousands of incoming emails on a daily basis. These inquiries and feedback can be automatically analysed to accelerate growth via optimising operational procedures and offering a product that solves customer problems in the most convenient way.

The classical methods of transforming textual data to a fixed structure are bag of words, n-grams and k-skip-n-grams. These methods have been thoroughly studied and despite their diverse shortcomings like high dimensionality, sparsity and putting semantic relationships aside, they generally ensure a strong baseline (Wang *et al.*, 2002: 1). To overcome the inexpedient dimensionality, an appropriate feature engineering process needs to be in place. This is achieved by either rescaling features, selecting relevant terms based on a statistical measure or combining redundant terms by generating synthetic frames (Sahlgren & Cöster, 2004: 2). There have been numerous initiatives at capturing the semantic meaning of words, e.g. latent semantic indexing as well as novel, neural network based methods which have proven to generally ensure a better performance than classical methods (Mikolov *et al.,* 2011: 608).

The variety of machine learning algorithms that have been applied in the textual domain is abundant. According to some authors (e.g. Liu *et al.,* 2017: 109), more complex methods almost consistently outperform simple ones. However, besides its classification capacity, the decisions made by the algorithm should be fairly easily interpretable to build trust in the end users. Furthermore, complex algorithms are often computationally heavier and with only little improvement on the classification performance, they may remain unjustified. To evaluate the potential of more complex algorithms, the author will employ extreme gradient boosting and stacked ensembling in addition to interpretable models like Naïve Bayes and logistic regression. If they significantly outperform interpretable models, then an effort could be made at interpreting their predictions. To benefit from novel representation methods like *word2vec*, the author will employ clustering via Euclidean and cosine distances.

Given thesis is with a practical nature and the purpose of it is to implement an automated text analysis model using data from TransferWise Ltd. that can be used to efficiently prioritise and measure incoming customer contacts. According to the purpose, the author has set up the following research tasks:

- give an overview of the nature and topicality of various business problems that can be solved using text mining methods;
- give an overview of text representation and feature engineering methods based on previous literature;
- give an overview of applicable modelling methods based on previous literature;
- give an overview of the dataset and research methodology;
- assemble various representations of the initial dataset and employ feature engineering;
- establish a classification model capable of prioritising and measuring incoming customer contacts.

Based on the purpose and research tasks, the structure of this thesis is divided into four principal parts – in the first part, the author describes the practical motivation and the usefulness of building automated text analysis tools. The second part provides a thorough theoretical overview of appropriate text representation, feature engineering and machine learning methods. In the third part, the author describes the data, its gathering principles and the general experiment setup and methodology. In the last part, the author establishes an optimal text analysis model, reports the progress of achieving it and evaluates its suitability qualitatively as well as quantitatively.

The thesis is important for both the company as well as its customers – handling customer inquiries based on their urgency or complexity ensures operational efficiency and a convenient customer experience, which is likely to accelerate growth. Furthermore, a clear understanding of the scope of various customer pain-points allows the company to make product related changes in a rapid manner. Besides its practical application, the author will generalise the results of numerous experiments and give an overview of the suitability and efficiency of various text mining techniques.

2 Practical motivation

To gain a competitive advantage, companies strive to make intelligent use of their existing data in decision-making. Arnott *et al.* (2017: 58) have stated that building efficient decision support systems is currently the largest area of IT investments in organisations. Traditional data mining methods have been efficiently applied on a variety of business tasks like financial fraud detection, revenue forecasting and product recommendation. However, most of these methods are applicable on structured data, leaving out by far the largest source of information in the form of text – it is estimated that nearly 80% of data in intranets and World Wide Web are in a textual form (Gupta *et al.,* 2009: 72). The process of automatically analysing large quantities of information in the form of written materials is an extension of data mining referred to as text mining. Harnessing written customer feedback and inquiries is a powerful addition to traditional decision support systems that promotes growth through being customer-focused and operationally efficient. The purpose of this subchapter is to map out and give an overview of various opportunities automated customer contact analysis offers for companies operating in the sector of financial services. The author will estimate the capacity and potential impact of these tasks and focus on solving one of them throughout the thesis.

Companies offering financial services receive thousands of **customer contacts** in a written form (e.g. emails) on a daily basis and in a standard set-up, they get handled on a first-in-first-out basis. However, this is not particularly efficient, as each contact has a different level of complexity and urgency. This raises the first opportunity – incoming customer contacts can be **prioritised** based on their **complexity** for the purpose of assigning more complex cases to more experienced agents. The latter is important for handling contacts in a more rapid and precise manner. Furthermore, as the number of contacts is somewhat seasonal (more payments are set up after paydays, which leads to more incoming contacts), then to avoid severe backlogs, contacts that are classified as "simple" by the algorithm, could occasionally be delegated to other teams besides customer support. To be scalable and cope with the increasing number of incoming contacts, businesses may benefit from **automatic question answering**, as simpler inquiries often do not need a personalised reply. This ensures that more complex inquiries get a precise reply in a timely manner. The author hypothesises that the complexity of an inquiry can be inferred by assuming that complex cases get a longer (in terms of handling time and reply length) and a more detailed reply from the company. However, measuring the efficiency of this approach is sophisticated and subjective, as there is no ground-truth to compare against, thus it is not in the scope of given thesis.

On the other hand, **urgency based prioritisation** of incoming contacts can be more direct. The majority of inquiries in given field are regarding a payment, hence the status of it can be a good label for assembling training data. The author hypothesises that contacts leading to cancelling the payment are more urgent and should be ranked higher in priority to avoid the cancellation. This is important due to numerous reasons, since handling an urgent inquiry in time is beneficial for the customer (ensures a better experience) as well as for the company (may result in a smaller cancellation rate). Furthermore, if after setting up a payment the customer sends an email which suggests that he/she has no intention on completing it, then an automated analysis system can be used to promptly take this into consideration for directing liquidity.

Aside from prioritising incoming contacts based on their urgency or complexity, a similar approach could be used for **measuring product related problems** – a precise overview of customer pain-points can help in prioritising product improvements. Based on prior expertise, product teams are mostly aware of their problems but struggle with quantifying them. This makes automated text analysis especially beneficial, as it could provide an unbiased and precise measurement of the scope of various problems.

Besides indirect ways, there are occasions where the dataset can be labelled directly – e.g. automatically **extracting customer sentiment** by using pre-labelled words. According to Jurafsky & Martin (2017: 74) the latter is relevant for every type of products. Furthermore, contacts with a negative sentiment could either be prioritised and attributed accordingly or used as measure of customer satisfaction.

Another common application of commercial text mining is **spam detection** – a binary classification task that is used to assign a label (spam or not spam) to emails. This is relevant and can become handy, since opening and instantly closing spam emails can take a lot of time in the big picture which could be contributed to solving actual customer problems. However, there are many built-in spam filters, thus it is not the most urgent problem to solve at the company level.

Jurafsky & Martin (2017: 74) have also pointed out the relevance of determining the characteristics of a text's author (e.g. gender, sex and native language). This is referred to as **authorship attribution** and based on prior domain knowledge, it can become handy in fighting financial crimes like fraud or money laundering. One of the methods money launderers use is performing payments in smaller batches to bypass the reporting requirements established by the law to not attract attention (Irwin *et al.,* 2011: 94). The reporting requirements usually specify a threshold and a timeframe such that payments exceeding them need to be reported to the authorities by the company. To bypass this, criminals create and operate multiple accounts. This is where authorship attribution can be beneficial – determining based on written text if profile holders are unique or if one person operates multiple profiles (which may indicate structuring). Although the suspicious patterns of many users with a similar profile activity and written language have been observed, then in most of the cases there simply might not be enough data, because contacting the company via an email is optional and not mandatory for using the product.

On the other hand, inferring characteristics like native language or age can be a useful addition to traditional **know your customer (KYC)** procedures by detecting mismatches between the inferred and provided information – e.g. if the customer has provided that he/she is 65 years old but written text resembles a youngster, then it may indicate fraudulent account takeover. The same applies for inferring the native language. Another beneficial characteristic to infer is stylometry – the way customers set up contacts (e.g. complexity of used words together and syntactic features) to detect possible account takeovers in case the written language changes significantly over customer lifetime. However, projects related to tightening KYC procedures are difficult to measure and their potential impact remains questionable.

Gupta & Lehal (2009: 62) have stated the potential of **text summarisation** – a task that aims to reduce the length of the document while retaining the most relevant ideas and overall meaning. The method is particularly useful if there are thousands of documents, out of which the reader would like to find the most relevant. In case of customer contacts, text summarisation can be used to build a deeper understanding of product-related problems since it can aid in choosing the most relevant emails for qualitative analysis. The latter is important for building products that solve actual customer problems in the most convenient way. Unlike prioritising and measuring product-related problems, the aim of this method is to guide people in solving problems revealed by the first.

Building insight about product-related problems can also be achieved using **concept linkage** – a method that connects related documents by shared concepts that people might not notice due to the large amount of data (Gupta *et al.,* 2009: 65). This can be used in addition to text summarisation, as it can aid a human in finding links to similar contacts more easily. However, evaluating these concepts is qualitative, which does not make it a suitable task for given thesis.

To encourage customer contact analysis, the author sees the potential of **feature extraction** – a task that aims to recognise and classify significant terms (e.g. names or relations) in a document using part of speech information and pattern matching (Dörre *et al.,* 1999: 399). In the financial sector, this could be used to automatically link contacts to corresponding payments via extracting relevant information (e.g. if the customer provides details like payment number) in case the contact can not automatically be linked (e.g. the email comes from an address that was not used for signing up). The latter is important as it can provide beneficial insights about the convenience of using the product via tracking metrics like contacts per transfer for various markets. Feature extraction could also be used to improve the efficiency of other operational procedures that rely on deriving information from texts. However, due to the specificity of the problem, the dataset needs to be manually annotated to extract references to payments, hence it is not in the scope of given thesis.

Based on the potential impact and due to the limitations on the capacity of given thesis, the author will hereafter focus on prioritising and measuring incoming customer contacts. Admittedly there are providers on the market that have expertise in solving similar tasks, but as emails often contain very confidential data (e.g. card details), then it is favoured to solve this task in-house. Furthermore, the accompanied expertise can be applied on any other tasks described above.

3 Theoretical background

3.1 Text representation

Textual data is rich in information, yet it lacks a fixed structure that is essential for applying classical machine learning algorithms. To establish an automated text analysis model, the data needs to be transformed into a structured form where the feature vector is numerical and with a fixed length. The purpose of this subchapter is to give an overview of both classical as well as novel methods that can be used to represent textual data.

3.1.1 Classical methods

Bag of words (a.k.a. vector space model) is one of the most common text representation methods, where documents are broken down (tokenized) to words that are mapped into a feature vector such that each dimension represents a distinct word and values indicate the frequency at which the word occurred in a specific document. As a consequence, the number of features can often be significantly greater than the number of training instances, which is likely to degrade the classification performance (Murphy, 2012: 18). Furthermore, as each document is made up of relatively few distinct words, then the resulting feature vectors are very sparse. Another shortcoming of given method is ignoring the order of word occurrences, thus discarding the semantic and locational information of words (Bernotas *et al.,* 2010: 218). Besides its shortcomings, it has ensured a strong baseline in various studies. The bag of words representation of a dataset consisting of the following sentences: (1) “I still have not received the money” and (2) “I have done the necessary” is presented in Table 1.

Table 1. Bag of words representation of a sample dataset.

	I	**still**	**have**	**not**	**received**	**the**	**money**	**done**	**necessary**
1.	1	1	1	1	1	1	1	0	0
2.	1	0	1	0	0	1	0	1	1

To tackle the aforementioned shortcomings, some authors have employed an extension called **n-grams** that aims to capture the information contained in adjacent words by considering sequences of *n* words instead of single words. Although the approach is theoretically justified, it generally does not outperform the standard bag of words (unigrams) method significantly – Apte *et al.* (1994: 237) observed that using only bigrams gives poor results, however, it might be useful for specific problems. Some authors (e.g. Wang *et al.*, 2002: 1) have efficiently used bigrams in addition to bag of words to enhance the classification accuracy. Therefore the author of given thesis will employ bigrams and trigrams as well as their combinations with bag of words, since according to Fürnkranz (2003: 1-2) n-gram sequences longer than 3 are not any more useful. The bigram representation of the sample dataset is presented in Table 2.

Table 2. Bigram representation of a sample dataset.

	I still	**still have**	**have not**	**not received**	**received the**	**the money**	**I have**	**have done**	**the necessary**
1.	1	1	1	1	1	1	0	0	0
2.	0	0	0	0	0	0	1	1	1

To overcome the sparsity problem accompanied by both of the aforementioned methods, Guthrie *et al.* (2006: 1225) proposed using **k-skip-n-grams** to efficiently model the coverage and context of documents. Unlike regular n-grams, this representation allows at most *k* (skip distance) words to be skipped while constructing the n-grams. D'hondt *et al.* (2012: 59, 62) observed that although using skip-grams improves the classification in terms of precision and recall, the accuracy tends to taper if skip-distance is greater than 2. Based on the work of D'hondt *et al.*, the author will only consider two-skip-bigrams, as it is seemingly a good trade-off between higher dimensionality and coverage of most important terms. The two-skip-bigram representation of the sample dataset is presented in Table 3.

Table 3. Two-skip-bigram representation of a sample dataset.

	I still	**I have**	**I not**	**still have**	**. . .**	**have necessary**	**done the**	**done necessary**	**the necessary**
1.	1	1	1	1	. . .	0	0	0	0
2.	0	0	0	0	. . .	1	1	1	1

The aforementioned methods can be used to construct a document-term matrix A where the entries a_{ik} measure the presence (e.g. frequency) of term *k* in document *i*. The simplest schemes for populating the matrix are boolean and term frequency weighting. As the number of term occurrences carries more information and both of the methods are similar in principle, then the author will discard boolean weighting. An appropriate scheme for populating the matrix can assist in overcoming the deficiencies of the aforementioned methods. In various applications (e.g. Sahlgren & Cöster, 2004: 5) weighting by the product of term frequency and inverse document frequency ($tf\text{-}idf$) has ensured the best classification performance. The $tf\text{-}idf$ of a term *k* in document *i* can be calculated as

$$tf\text{-}idf(i,k) = f_{ik} \times log\,\frac{N}{df_k} \;,$$

where *N* is the number of documents, df_k is the number of documents where term *k* occurs and f_{ik} is the frequency of term *k* in document *i* (Jurafsky, 2017: 278). Therefore $tf\text{-}idf$ assigns higher weights to terms that occur frequently in a specific document, but rarely in other documents and lower weights to terms that occur in many documents overall. However, $tf\text{-}idf$ might not be sufficient as it does not take into consideration that documents can be of different length. As the length of incoming emails can differ significantly, then based the work of Salton & Buckley (1988: 518) the author will use cosine normalised $tf\text{-}idf$ (referred to as $\boldsymbol{tfc}$) that can be calculated as

$$tfc(i,k) = \frac{tf\text{-}idf(i,k)}{\sqrt{\sum_{t=1}^{M}\left[f_{it} \times log\left(\frac{N}{df_t}\right)\right]^2}} \;,$$

where f_{it} is the frequency of term *t* in document *i*, df_t is the number of documents where term *t* occurs and *M* is the number of terms. Concisely, the denominator is the length of the vector of document *i* $tf\text{-}idf$ values.

3.1.2 Novel methods

As machine learning techniques have developed rapidly over the recent years, then neural network based models have proven to outperform the aforementioned classical methods

(Mikolov *et al.,* 2011: 608). The research around novel representation methods has mostly focused on capturing the semantic meaning of terms.

According to Goldberg & Levy (2014: 1), one of the most prevalent methods that provides state of the art word embeddings through capturing the semantic meaning is ***word2vec***. The model learns a high dimensional vector representation of words such that semantically similar words are close to one another in the vector space. This is achieved by formulating the training process as a supervised problem, where a neural network with one hidden layer is trained to predict a word or the surroundings of a word based on some input that is parametrised by the window size. The obtained weights between the input and the hidden layer can then be used as word embeddings. The most prevalent architectures for efficiently learning a distributed representation of words are **skip-gram** and **continuous bag of words (CBOW)**. In the first case, the network tries to predict the surrounding words given an input word. Inversely, making use of the CBOW architecture, the network tries to predict a target word from its surroundings. Mikolov *et al.* (2013: 5) observed that the CBOW architecture works better on both syntactic and semantic tasks. Both of the architectures are illustrated in Table 4. Apropos, the window size in both examples is 2 and the sample sentence is “Where is my money?”.

Table 4. Difference of CBOW and skip-gram architectures on a sample sentence.

Method	Training sample(s)	Target variable(s)
CBOW	1. [is,my,money] 2. [where,my,money] 3. [where,is,money] 4. [is,my]	1. where 2. is 3. my 4. money
Skip-gram	1. where 2. is 3. my 4. money	1. [is,my] 2. [where,my,money] 3. [where,is,money] 4. [is,my]

According to Mikolov *et al,* (2013: 7), *word2vec* models require a large vocabulary (size in billions) for training an accurate model, thus the domain-specific corpus is not sufficient and the author will use a pre-trained model published by the same authors at Google. The model was trained on Google News corpus of nearly 100B words with a hidden layer of 300 neurons (Google Code Archive).

However, *word2vec* modes learn a representation for each word, not the whole document. Mikolov & Le (2014: 1188) have proposed using a similar method called paragraph2vec that learns a dense, fixed-length vector representation of documents with varying length. Unfortunately interpreting and qualitatively analysing document level embeddings learned by neural networks is complicated (Kim *et al.,* 2017: 336). Furthermore, *paragraph2vec* models trained on a different context can not be generalized similarly to *word2vec* and as there is not enough data for training, then this method is discarded. There are multiple ways on how to obtain document representations from word embeddings. The simplest method is coordinate-wise aggregation (e.g. taking maximum, minimum or average) of a document's word embeddings. Some authors have also proposed using the average of word vectors weighted by $tf-idf$ (e.g. Lilleberg *et al.,* 2015: 136), but have not observed any improvement in classification accuracy. The shortcoming of these approaches is discarding the semantic information of words that is obtained by using *word2vec*. To make use of the semantic relationships and tackle the problem of synonymy, similar words can be divided into *K* clusters that can be used to construct any of the aforementioned classical representations. Kim *et al.* (2017: 344) referred to this as **bag of concepts** and showed that compared to a regular bag of words, latent semantic analysis and *word2vec* averaging, it ensured a significantly better classification accuracy. Furthermore, the concept clusters are interpretable, which makes it a suitable method in given context.

3.2 Feature engineering

As the dimensionality of aforementioned methods is dependant on the number of terms, then it can get unreasonably large. Kohavi & John (1997: 275) have pointed out that many machine learning algorithms deteriorate in terms of accuracy if there are too many features (of which not all might be relevant). Therefore the obtained representations are often refined before used in machine learning algorithms. The process is referred to as feature engineering and according to Sahlgren & Cöster (2004: 2), it consists of feature selection and feature extraction.

3.2.1 Feature selection

Feature selection is a task that aims to reduce the dimensionality of the dataset by discarding irrelevant terms. According to Sebastiani (2002: 16) this is mostly done by using wrapper or filtering methods. The wrapper approach is used to construct a feature set by training classifiers on various feature subsets. Although it has an advantage of combining a feature set that fits best for a specific algorithm, it is computationally intractable and hence will not be considered in given thesis. Filtering is an approach where features are selected based on a statistical measure. Yang & Pedersen (1997: 412) observed in their comparative study of various feature selection methods on textual data that information gain and chi-square are the most effective metrics in terms of classification accuracy. These methods rely on the assumption that best terms for classification are the ones that are distributed most differently between classes. At the same time, they admitted that document frequency thresholding performs relatively well and has the lowest computational cost.

Information gain is a metric that indicates how many bits of information the presence of a word entails (Jurafsky & Martin, 2017: 99). A more intuitive definition states that information gain measures the reduction in entropy of the target variable after a specific term has been observed. The most useful features are the ones that reduce the entropy the most, by occurring more frequently in one class than others. The information gain (*G*) of a term *k* can be calculated as

$$G(k) = -\sum_{i=1}^{C} P(c_i) \times log\, P\,(c_i) + \, P(k)\sum_{i=1}^{C} P(c_i|k) \times log\ P(c_i|k)$$
$$+ \; P(\bar{k})\sum_{i=1}^{C} P(c_i|\bar{k}) \times log\ P(c_i|\bar{k})\,,$$

where c_i is the *i-th* class, k and $\bar{k}$ correspondingly indicate if a document does or does not contain term *k* (Yang & Pedersen, 1997: 415). This method has been widely used in similar contexts, thus it will also be considered as one of the feature selection metrics in given thesis.

Chi-square statistic is used to measure how the results from an observation differ from the expected results. The chi-square value of term k in class c is defined as

$$\chi^2(k,c) = \frac{N \times (AD - CB)^2}{(A + C) \times (B + D) \times (A + B) \times (C + D)},$$

where N is the total number of documents, A is the number of times term k occurs in class c, B is the number of times term k occurs outside class c, C is the number of times term k does not occur in class c and D is the number of times neither term k or class c occur (Yang & Pedersen, 1997: 415). The aforementioned occurrences are summarised in Table 5.

Table 5. Contingency matrix for calculating chi-square statistic.

	$\boldsymbol{c}$	$\boldsymbol{\bar{c}}$
$\boldsymbol{k}$	A	B
$\boldsymbol{\bar{k}}$	C	D

Concisely, the statistic is equal to 0 if term k and class c are independent. Therefore terms with a low chi-square value should be discarded as they have a low predictive value. The author of given thesis will rely on the work of Khoo *et al.* (2006: 22) and only consider words with largest chi-square values. However, one must take into account that the statistic is known to be not reliable for low-frequency terms as a high significance may be detected for features with a low effect on classification.

Document frequency thresholding is a simple yet effective feature selection method where terms that occur less than a predefined threshold are removed from the feature space. Despite the fact that the method is simple and its computational complexity scales almost linearly, it is not considered to be a principled approach (Yang & Pedersen, 1997: 413). Although rarely occurring terms are often considered to be informative, the same authors (1997: 419) showed in a number of experiments that it is possible to remove nearly 90% of terms without making concessions on classification accuracy. However, if the distribution of term occurrences has a long tail of rare terms, then there is a risk of discarding features that jointly may cover a large proportion of the documents. Unlike aforementioned methods, it can be computed without knowing the class labels, thus it is an appropriate method to consider in given thesis.

There is a large variety of feature selection metrics, but as they generally perform similarly (Khoo *et al.,* 2006: 22), then the author of given thesis sees information gain, chi-squared test and document frequency thresholding as sufficient.

3.2.2 Feature extraction

Feature extraction is a method that is used to combine redundant terms (e.g. synonyms) or terms that need distinction (e.g. homonyms) by generating synthetic terms using term clustering or latent semantic analysis (Sebastiani, 2002: 18).

Term clustering is a method that aims to group together terms that are semantically similar and according to Lewis (1992: 38), the clusters are formed by terms that tend to co-occur in documents. However, there have not been any major achievements in using unsupervised term clustering and the usefulness remains questionable, as the clusters are most often formed randomly (*ibidem*). Slonim & Tishby (2001: 8) encouraged to use supervised clustering instead – they reported a higher accuracy on various datasets by forming word clusters using information bottleneck method. However, applying additional transformations increases the latency and with little or no improvement on classification capacity, it remains unjustified and will be discarded in this form for now.

Latent semantic analysis (LSA) is a method proposed by Deerwester *et al.* that maps a document-term matrix into a semantic space and uses singular value decomposition to generate orthogonal factors that can be arranged so that less important influences are ignored (1990: 391). As a result of this mapping, the new dimensions are a linear combination of original features, which apropos are not anymore interpretable. The method infers the theme of a document by analysing word co-occurrences (relying on the assumption that words with a similar meaning occur in similar documents) and has been proven to improve the classification accuracy significantly (Schütze *et al.,* 1995: 2, 8). Based on Sebastiani (2002: 20), the shortcoming of given method is that if a term is very good itself, then its discriminative power may be lost in the new feature space.

3.3 Classification algorithms

Text classification (a.k.a text categorization) is a supervised learning task which aims to assign unseen documents to pre-defined classes by learning from a training set of labelled documents. There is a large variety of classical algorithms that have been applied in text classification – according to Desmet & Hoste (2018: 66), one of the most prevalent algorithms are **Naïve Bayes** and **support vector machines**. In addition to this, Aggarwal & Zhai (2012: 165) have stated that text classification is often performed using **decision trees**. In a recent edition of Speech and Language Processing (Jurafsky & Martin, 2017: 75, 92), the authors pointed out the common use of **neural networks**, **logistic regression** and **random forests** for text analysis.

3.3.1 Algorithm selection

As the set of algorithms employed in previous literature is abundant and their performance dependant on the specifics of the corpus (e.g. document type or capacity), then comparing them all is not in the scope of this thesis. Furthermore, their performance is often reported on disparate datasets, thus they can not be directly compared. Therefore the author has set up a relevant selection criteria – besides its classification capacity, the decisions made by the algorithm should be fairly easily interpretable. The latter is admittedly important in given context for building trust in the end users, as predictions without a clear justification are not sufficient to act upon. However, if a more complex model significantly outperforms simple ones, then an effort could be made at producing interpretable results (e.g. via employing proxy models). As the training process is not time-critical, then the author has not set up any restrictions on the computational complexity. The purpose of this subchapter is to evaluate the suitability of forementioned prevalent algorithms for given task and select a reasonable subset of them for further experiments. The selected algorithms will be more thoroughly analysed in the next subchapter.

According to Ikonomakis *et al.* (2005: 971), support vector machines generally provide poor results in terms of recall, which due to the nature of given research problem is not advisable. The same condition was observed by the author while conducting the initial experiments. Similarly, as single decision trees can be very unstable and sensitive, then they will be discarded. Mar & Naing (2008: 514) have stated that figuring out the optimal architecture for neural networks is difficult and as the results are hard to interpret, then it is not in the scope of given thesis.

Naïve Bayes has been applied in a variety of text categorisation problems. Although the naïve assumption implies that features are equally important (which in turn may result in biased posteriors), many authors (e.g. Hastie *et al.,* 2001: 211) have stated that the model often outperforms more complex ones. Furthermore, its naïve assumption makes it computationally efficient, which is beneficial as the number of experiments to be carried out is relatively large. Logistic regression has similarly been extensively applied in the textual domain due to its interpretability and efficiency. It is particularly useful for modelling textual data due to a built-in regularisation.

Random forest is an ensemble classifier proposed by Breiman (2001: 29) that builds numerous weak decision trees on random subsets of data such that each single tree is built using a random subset of features. It is proven to often ensure a better performance than support vector machines or neural networks (Breiman, 2001). Based on previous experience, the author acknowledges that building single trees in an additive manner instead of independent trees (like random forest) is more efficient. This process is referred to as gradient tree boosting, as each subsequent tree tries to correct the mistakes made by the previous tree. Gradient tree boosting has been widely used in various applications to achieve state of the art results. Liu *et al.* (2017: 109) used *xgboost* implementation of gradient boosting in a textual domain and reported that it almost consistently outperformed random forests, decision trees and logistic regression, which makes it a suitable model to consider.

Zhu *et al.* (2015: 2) have stated that a single classifier is often not optimal for large, heterogeneous problems and have proposed to use multiple classifier systems (MCS) to improve the accuracy. Although the approach of combining strong learners is counterintuitive compared to the studies of ensembling weak learners (e.g. random forests), Laan *et al.* (2007: 17) proved that using a super learner (a.k.a stacking) often ensures a better discriminative performance.

Concisely, the author will employ interpretable models such as logistic regression and Naïve Bayes and use more complex models like xgboost and stacking to ascertain if more complex models significantly outperform simple ones and an effort should be made to put them in practice.

3.3.2 Interpretable models

Naïve Bayes is a simple and computationally efficient probabilistic classifier that has been efficiently applied in a variety of natural language processing tasks. The algorithm is derived from Bayesian inference, in which the class of a document is predicted by considering the product of prior class probability and the likelihood of a document. However, as there is a large number of possible feature combinations, then the calculation of conditional probabilities becomes intractable. To overcome this, the algorithm makes a naïve assumption on conditional independence when calculating class probabilities, such that the probabilities could simply be multiplied as

$$C = argmax_{c \in \mathrm{C}} P(c) \prod_{i \in \text{vocabulary}} P(w_i|c) \,,$$

where C is the assigned class, $P(c)$ the percentage of documents in the training data from class *c* and $P(w_i|c)$ the fraction of how often word w_i appears among all other words in class *c* (Jurafsky & Martin*,* 2017: 77). Unfortunately the latter imposes a problem because estimating the likelihood of a word that does not occur in class *c* (but occurs in others) results in $P(w_i|c) = 0$ and since the probabilities are naively multiplied, then the probability of class *c* is equal to zero despite other indicators. A simple way to overcome this is to use Laplace smoothing by incrementing the numerator and denominator to achieve non-zero probabilities as

$$p(w_i|c) = \frac{count(w_i, c) + 1}{(\sum_{w \in \mathrm{V}} count(w, c)) + |V|},$$

where $|V|$ is the size of the vocabulary (Jurafsky & Martin, 2017: 47). According to Peng *et al.* (2003: 320) Laplace smoothing is usually not efficient. Another way to overcome zero probabilities is by specifying and assigning a minimum value to probabilities less than a predefined threshold. Based on the previous empirical literature, the author will assemble a grid of the following parameters – minimum probability, threshold and Laplace smoothing parameter. The algorithm is particularly suitable for the scope of this thesis, as it provides great justification behind its predictions – the importance of a word could simply be inferred by comparing the posterior probabilities by classes.

Logistic regression is a prevalent discriminative method used for modelling problems where the target is a categorical variable with two categories. Similarly to Naïve Bayes, it is a linear classifier – the posterior probabilities are modelled via a linear combination of input features (Hastie *et al.,* 2009: 119). To squash the linear outputs to a legitimate probability range, a logistic function in the form of

$$p(c|x) = \frac{e^{x^T\beta+\beta_0}}{1+e^{x^T\beta+\beta_0}},$$

is applied, where β is the weight vector, β_0 the bias term (intercept) and x the input features (Hastie *et al.,* 2009: 119). The weights of the model are learned using conditional maximal likelihood estimation – the weights are chosen with the purpose of maximising the probability p of class c given input features x (a.k.a $p(c|x)$). The objective function can be defined as a penalized likelihood

$$\frac{1}{N}\sum_{i=1}^{N}\left(y_i(x_i^T\beta+\beta_0)-\log\left(1+e^{x_i^T\beta+\beta_0}\right)\right)-\lambda\left(\alpha||\beta||_1+\frac{1}{2}(1-\alpha)||\beta||_2^2\right),$$

where the *N* is the number of observations. The second part of the objective function stands for penalisation using a combination of Lasso and Ridge regression (a.k.a elastic net penalty) which is important to avoid overfitting (Jurafsky & Martin, 2017: 97, Hastie *et al.,* 2009: 73). In the first case, the ℓ_1 norm of weights is penalised, which results in a sparse solution. As many weights will be equal to zero, then it is a good method for feature selection. Ridge regression on the other hand penalises the ℓ_2 norm of the weights (they are proportionally shrinked), which ensures that the weights do not get too large. The λ parameter controls the amount of penalisation applied, whereas α controls the distribution between ℓ_1 and ℓ_2 penalisation. Given optimisation task is a convex problem (Jurafsky & Martin, 2017: 97), thus methods like gradient ascent can efficiently be applied. In this case, the weights are updated based on the partial derivative of the objective function with respect to the weights. To find the optimal model, the author will assemble a grid of λ and α values.

3.3.3 Complex models

XGBoost (extreme gradient boosting) is a highly efficient tree boosting implementation that is considered to be the de-facto choice of ensemble methods that has been applied in various tasks including natural language processing (Chen & Guestrin, 2016: 785). The algorithm employs gradient boosting by building numerous models in an additive fashion on the gradient of the loss function being minimised (Friedman, 2002: 367). The set of *K* additive functions is learned by minimising the regularised objective that is defined as

$$\mathcal{L}(\phi) = \sum_{i} l(\hat{y}_i, y_i) + \sum_{k} \Omega(f_k)\,,$$

where l is a differentiable loss function (e.g. logloss) that measures the differences between predicted $\hat{y}_i$ and true values y_i and Ω is a penalisation term of model complexity (Chen & Guestrin, 2016: 786). The predicted values can be obtained by using all *K* trees as

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i)\,,$$

where f_k is the prediction from *k-th* decision tree and x_i the input features of *i-th* observation. The additive learning starts from a constant prediction and adds a new function f_k at each iteration. The complexity penalisation Ω for each of the *k* trees is defined as

$$\Omega = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2\,,$$

where the complexity of each tree is penalised in regards to the number of leaves (*T*) and the scores (*w*) on leaves by parameter γ. The tree structure is then obtained in a greedy manner by using information gain as a split criterion.

It has been shown (Friedman, 2002: 367) that the accuracy, risk of overfitting and speed of training can be significantly improved by considering a random subsample of training data. Similarly, Chen & Guestrin (2016: 787) stated it is beneficial to consider a random subsample of features for additive model and shrink the weights to avoid overfitting. The latter is accomplished by scaling the weights after each iteration by a learning rate coefficient. It is recommended to set the learning rate as low as possible, yet this may result in a slower convergence, because the influence of previous trees is reduced and more trees may be needed. As single trees are learned in a greedy manner, then to avoid overfitting, the author will set different limits on their depth. Concisely, the author will combine a grid of the following parameters to tune – learning rate, column sampling rate, row sampling rate, the number of trees and maximum tree depth. Although the model is not easily interpretable, it will be used in given thesis due to its capacity that has ensured the first place at numerous machine learning competitions.

Stacking is a process that includes training a second level learner on top of base learners. Da Silva *et al.* (2014: 178) showed that combining classifiers trained on diversified components can provide state-of-the-art results on various textual datasets. The same authors have stated that for an efficient ensembling procedure, the learners should be as diverse as possible to cover a prediction space as large as possible. Therefore the author will use stacking to combine strong and diverse classifiers into one ensemble via logistic regression (as learning from just a few features does not require a more sophisticated approach). This also implies that the same parameters and principles described above apply to the super learner as well as base learners.

3.4 Clustering algorithms

To apply any of these supervised algorithms on *word2vec* embeddings, the words that appear in similar context will be grouped together to assemble a bag of concepts representation. The terms are grouped via clustering, which according to Zhang & Zhou (2004: 514) is an unsupervised task that aims to group similar observations into clusters while maximising the inter- and minimising the intracluster similarities. According to Passalis & Tefas (2017: 277), term vectors are most often grouped using k-means clustering such that the centers are used as new features. The algorithm works in an unsupervised and iterative manner by initialising *K* cluster centers (μ) and assigning observations (*x*) to a closest cluster (*k*) using Euclidean distance as

$$z_i^* = argmin_k ||x_i - \mu_k||_2^2 \,.$$

After the reassignment, the cluster centers are updated as

$$\mu_k = \frac{1}{N_k} \sum_{i:z_i=k} x_i \,,$$

where N_k is the number of observations in cluster *k* (Murphy, 2012: 352). The process is repeated until no observations will be a reassigned. The initial cluster centers can be obtained in numerous principled ways, but due to their computational complexities, the author will set the initial centers on a random basis. This process will be repeated multiple times to ensure that the model converged to a global optimum.

However, according to Huang (2008: 51), Euclidean distance performs poorly on textual data and one of the most popular similarity measures applied is cosine distance that is defined as

$$\cos(x, y) = \frac{\sum_{i=1}^{N} x_i y_i}{\sqrt{\sum_{i=1}^{N} x_i^2}\sqrt{\sum_{i=1}^{N} y_i^2}},$$

where *N* is the dimensionality of the word embedding and x_i and y_i the value in *i-th* position of the corresponding word embedding. The cosine similarity of words *x* and *y* can be calculated using the inner product, therefore the measure is between minus one and one, where larger values indicate greater similarity and zero indicates decorrelation (orthogonality). This type of clustering is also referred to as spherical k-means (Dhillon & Modha, 2000: 149). Cosine similarity is particularly useful in given context, as according to Jurgovsky *et al.* (2016: 203) arithmetic operations on *word2vec* representations accurately reflect the semantic and syntactic properties, which makes the angle more relevant than the magnitude.

Concisely, the author will employ will employ unsupervised algorithms like k-means and spherical k-means to make aforementioned supervised algorithms applicable on *word2vec* embeddings.

4 Experiment setup

The empirical part of given thesis relies on a corpus that is compiled by the author using data from TransferWise Ltd. The raw corpus consists of ~13000 most recent incoming customer contacts in English. As the dataset contains sensitive and confidential information, then the contents (as well as the exact class distributions) can not be disclosed. The documents are annotated by the customer at the time of submitting the inquiry through a special contact form. The form includes a list of predefined categories that cover a variety of issues (e.g. questions around new products, verification, funding the payments etc.) that customers, once logged in, can choose from. However, not all annotations may be correct as the categories are rather generic and customers may not be familiar with the scope of them. To verify that we can be confident in labels provided by the customer, the author manually went over a relatively large subset of contacts.

The corpus was compiled by following the principles of stratified sampling – the ratio of contacts from other categories is alike. The classification problem is set up as one-vs-all – the author chose an issue with the highest potential and urgency and labelled all other categories as not related to this specific problem. The size of given corpus is limited to ~13000 documents due to a large number of experiments carried out using various methods described in the first chapter. The limit was set by gathering all contacts in the previous N days. It is estimated that nearly half of all incoming contacts are through the form. Based on prior expertise, the author is certain that there is no heavy bias between form and non-form contacts (e.g. in topics or quality of text) and as the issues can be ranked based on their urgency and complexity, then it is an excellent way for gathering training data.

The optimal model will be obtained by employing classical as well as novel text representation methods combined with appropriate weighting schemes and carefully tuned classifiers. As the author has access to all contacts, then the optimal model together with corresponding feature representation, engineering and weighting method will be put into practice using a significantly larger corpus by considering a longer look-back period. Once the model is put into practice, then to enhance its performance further, an operational procedure for relabelling the contacts can be implemented. The latter implies that wrong predictions get marked accordingly to avoid similar mistakes in the next training iteration. The applicability of the final model will be evaluated quantitatively as well as qualitatively.

Before converting the corpus to any of the representations described in subchapter 1.2, the author will apply the following common filters:

- lower case conversion – the case of a term is mostly irrelevant, thus words will be converted to their lower case to be interpreted alike;
- removing stopwords due to their very low discriminative value;
- grouping domain-specific words – the author has compiled a list of words that should be interpreted alike (e.g. names of partner banks, dates, reference numbers etc) that are concatenated into unified tokens;
- removing words occurring in less than 5 documents, as they rarely carry any discriminative value. Along with this – email addresses, links etc. are removed;
- removing punctuation, whitespaces, numbers and non-ASCII characters.

After filtering, the number of words was reduced from ~20 000 to ~3000. This magnitude indicates the importance of filtering before applying any further transformations. The obtained training corpus is summarised in Table 6. Similarly, after converting the corpus to aforementioned classical representations, the author obtained ~8500 bigrams, ~2500 trigrams and ~25000 two-skip-bigrams.

Table 6. Summary of the corpus after filtering.

Number of documents	~13000
Document type	Incoming customer contacts (in English)
Annotation	Contacts related to a specific problem
Annotation type	Boolean (one-vs-all), annotated by customers using a drop-down of predefined values
Class distributions	20% related to problem of interest, 80% all others
Sampling strategy	Most recent contacts; similar distributions amongst other contact types

The classification capacity will be evaluated on a confusion matrix, which assembles all predictions made by the model together with their actual labels. Based on this, false positives (FP) and false negatives (FN) can be distinguished. In the first case, the model classifies a contact to be related to the problem of our interest, whereas actually it is not. In case of false negatives, the inverse is true – the model classifies a contact to not be related to a problem of our interest, whereas it actually is. According to Ikonomakis (2005: 972), a common metric considered in evaluating text classifiers is accuracy, which indicates the overall ratio of correct decisions. However, it might not be suitable for this specific task due to the unbalanced dataset, as a high accuracy could be achieved by just assigning each document to the majority class. Sebastiani (2002: 37) has stated that precision and recall are one of the most common metrics used to evaluate the efficiency of text classification. Precision (a.k.a positive predictive value) is the fraction of true positives out of all positive predictions. This essentially indicates the fraction of contacts that is actually related to the problem of our interest out of all the contacts that the model predicts to be related. This is one of the most important metrics for adequately measuring product-related problems, however, a high value could be achieved by simply predicting the class of interest rarely. To make sure the model can retrieve most of the relevant contacts, the author will use recall (a.k.a sensitivity). This metric essentially measures the fraction of contacts related to the problem of our interest that the model was able to recall. Gu *et al.* (2008: 1023) have stated that precision and recall solely may not be sufficient, therefore the author will use their harmonic mean (F1-score). As models with similar F1-scores can differ remarkably in their applicability, then similar models will additionally be compared based on their precision and recall. Thus, F1-score is used to solely rank numerous models, out of which the better-performing ones will be analysed more thoroughly.

Fawcett (2003: 3) has stated that the analysis of receiver operating characteristic (ROC) curve is an efficient method to evaluate classification models on unbalanced datasets as it is a good trade-off between true positive and false positive rates. The author will use ROC curve analysis to take into consideration the class imbalance – the performance will be evaluated using the area under the ROC curve (AUC) which is always between zero and one (where greater values indicate a better model). The intuition behind constructing a ROC curve lies in ordering the predictions by probabilities and moving step-by-step upward (if the decision is correct) and to the right (if the decision is incorrect) from the origin of coordinates. Therefore the AUC value for a model which makes no mistakes will be 1, as the curve will be constructed by only moving upwards. The latter implies that the false and true positive rates are correspondingly equal to 0 and 1. On the other hand, AUC value 0.5 is considered as a random baseline. Furthermore, analysing the ROC curve can provide useful insights about the behavior of different models and can aid in finding the optimal decision threshold that keeps false and true positive rates at an acceptable level to make the model applicable in practice. The principle of ROC curve is illustrated on Figure 1.

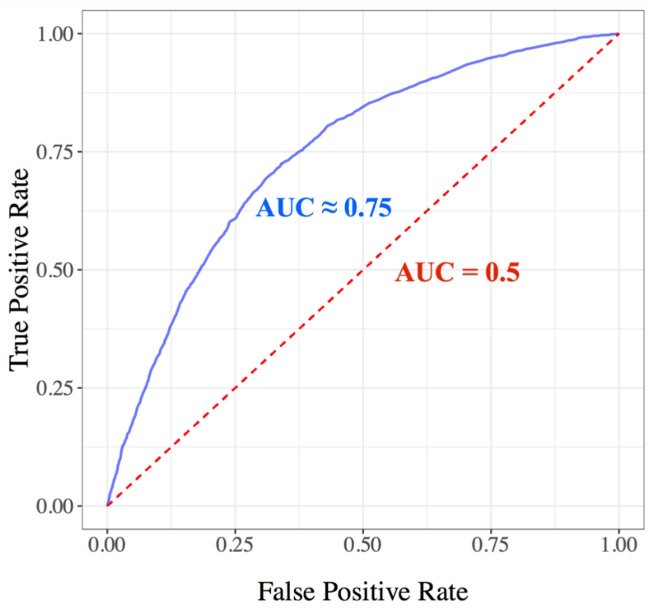


Figure 1. Principle of ROC curve analysis.

The performance of a model is traditionally evaluated by splitting the dataset into two – one that is used for training and one that is used for testing. It is important to evaluate the model on unseen data, otherwise the results will be too optimistic. According to Kuhn & Johnson (2013: 77-78) it may not be sufficient to evaluate the performance on a single holdout set, as the results may be biased. Furthermore, as the implementation is carried out iteratively, then there is a risk of overfitting towards the holdout set and obtaining a model that does not generalize well in practice. Kohavi (1995: 1145) observed that stratified 10 fold cross-validation has the lowest bias and variance compared to other cross-validation schemes and bootstrapping. Therefore the author will split the data into 10 mutually exclusive subsets that follow similar class distributions. Thereupon 90% of the data is used to train 10 models, leaving out a different 10% for validation at each time and the overall goodness of models will be estimated by averaging the F1-scores and AUC values obtained on corresponding holdout sets. The ROC curves for cross-validated models will be visualised by aggregating sensitivity and specificity pairs over every fold and prediction.

The stacked model will be trained and evaluated in a similar manner – the author will first train a set of strong base learners (one for each model type) using grid search and stratified 10 fold cross-validation while also saving the prediction probabilities on validation folds. These predictions will then be used to construct a new NxL training frame, where N is the number of rows in the training set and L is the number of base learners. This data is used to implement a logistic regression based meta-learner, using the same cross-validation scheme as proposed above.

The plentiful set of methods that the author will employ is summarised on Figure 2.

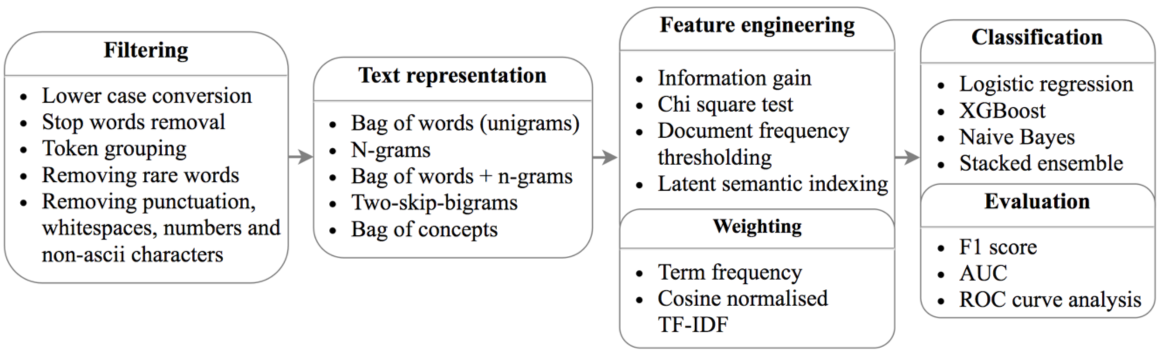


Figure 2. Overview of applicable methods.

Concisely, the author will train numerous model combinations using various text mining techniques to establish an automated text analysis model that can be used to prioritise and measure incoming customer contacts.

5 Results

5.1 Classical methods benchmark

The purpose of this subchapter is to establish a performance benchmark by using classical text representation methods. To achieve this, the author will train numerous models on each representation by considering every combination of aforementioned classification, feature engineering and weighting methods. Furthermore, the author will establish baselines for every model by using the complete dataset.

Bag of words

The progress of obtaining an optimal model using a bag of words representation combined with TFC weighting is presented on Figure 3.

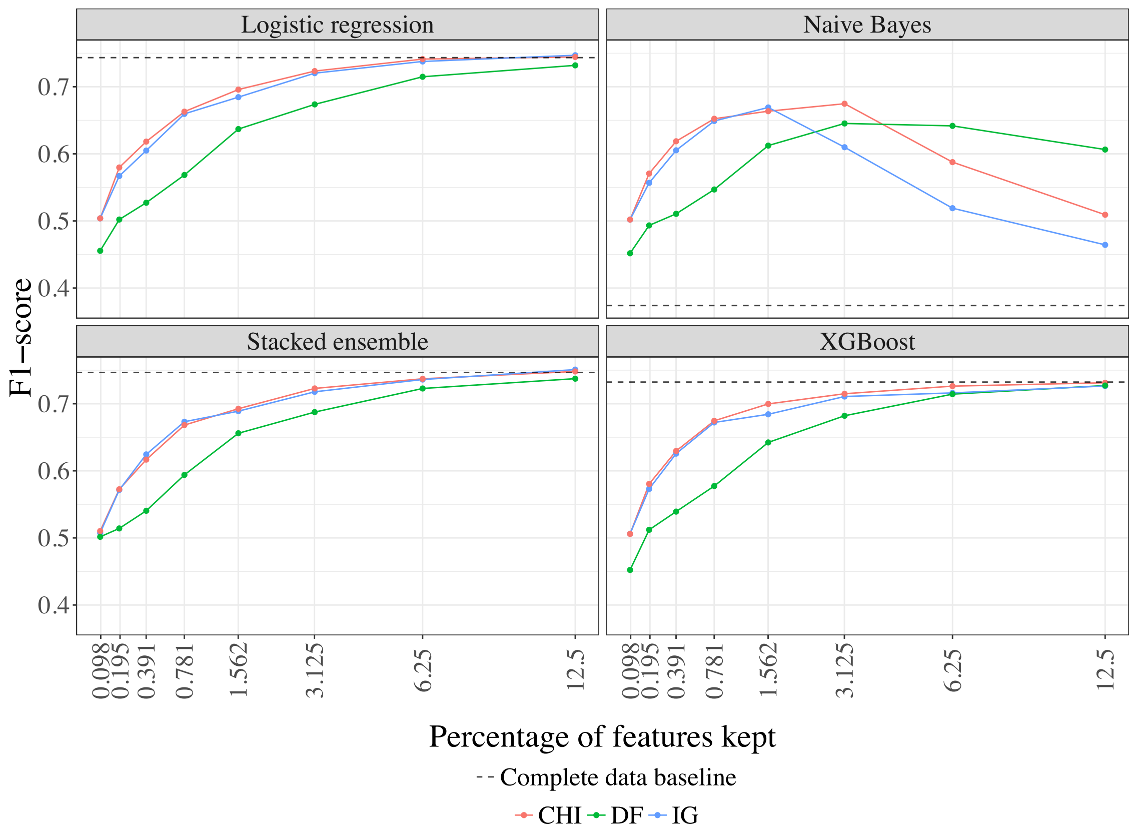


Figure 3. Unigram classifier performances by feature selection method using TFC weighting.

It is evident that using information gain and chi-square test ensure nearly similar results for every classifier. On the other hand, document frequency thresholding tends to perform slightly differently but is generally able to converge to the same optimal value, although at a slower pace. For methods other than Naïve Bayes, the convergence to baseline is achieved by using only 12.5% of original features. However, the incremental improvement is marginal and in case of computational limitations, using solely ~1% of the original features ensures more or less the same performance. Furthermore – the author reckoned that the performance did not improve any further by considering more than 12.5% of features. In case of Naïve Bayes classifier, the performance on complete data is worse than using only a fraction of features seemingly due to the absence of regularisation in the algorithm. The best performance for the latter was achieved by selecting 3.125% of features with the highest chi-square values. Furthermore, the models tend to perform rather alike on various feature subsets. The same progress using TF weighting is presented in Appendix 1 due to the similar results. The optimal performance for each classifier and weighting scheme is summarised in Table 7.

Table 7. Optimal unigram model performance by weighting method.

		F1	**AUC**	**Feature selection metric**	**% / number of features chosen**
Naïve Bayes	TF	0.67	0.847	Chi-square	3.125% / ~95
	TFC	0.674	0.84	Information gain	12.5% / ~375
Stacked ensemble	TF	0.744	0.919	Chi-square	12.5% / ~375
	TFC	**0.75**	0.925	Information gain	
XGBoost	TF	0.735	0.917	None	100%
	TFC	0.732	0.912		
Logistic regression	TF	0.739	0.915	Information gain	12.5% / ~375
	TFC	0.747	**0.927**		

The optimal model in terms of F1-score was obtained using a stacked ensemble. On the other hand, the greatest AUC value was achieved by logistic regression. Unlike previous studies, complex models like XGBoost and stacked ensembling did not outperform logistic regression considerably. Likewise, using a more sophisticated weighting scheme seemingly does not have any significant influence on the performance using bag of words representation. An F1-score of 0.75 is a relatively strong baseline – in that instance, the recall and precision were correspondingly 0.721 and 0.782. Thus the model was able to detect ~72% of relevant cases (related to the problem of our interest) and ~78% of its relevant class predictions were correct. On the other hand, the superiority is insignificant compared to logistic regression, which had a recall and precision value correspondingly of 0.794 and 0.706. It is evident, that although the models have similar F1 values, they have a different behavior. In given context, the author sees logistic regression more suitable due to its interpretability and an advantage of detecting ~7% more contacts of our interest.

Bigrams

The author repeated the same analysis using a bigram representation. The progress on Figure 4 is reported using TFC weighting and the initial results of using TF weighting are similarly presented in Appendix 2. In case of bigrams, the complete data baseline, as well as the optimal model was achieved using only 6.25% of features instead of 12.5% as in the bag of words approach. However, as there are nearly three times more bigrams, then the actual number of features used was greater.

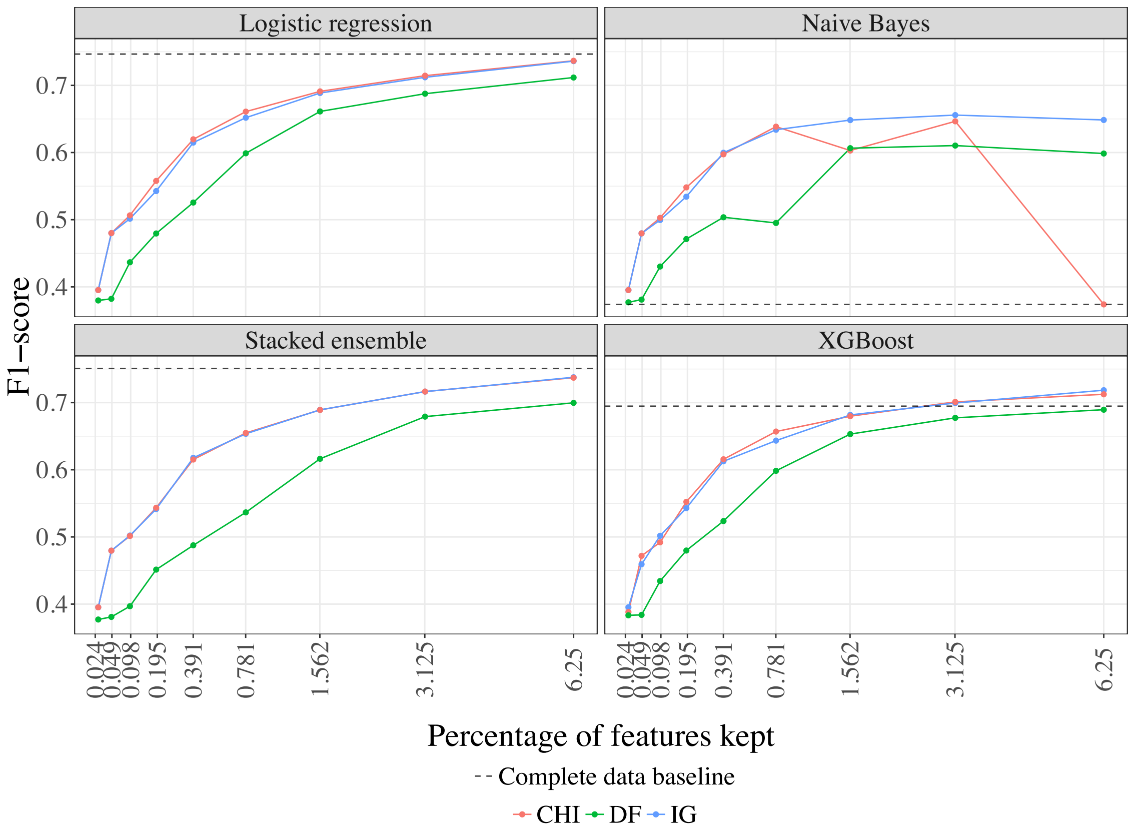


Figure 4. Bigram classifier performance by feature selection method using TFC weighting.

The above-presented figure confirms that supervised feature selection methods ensure a similar performance. Furthermore, it is occasionally even impossible to distinguish their performance (e.g. in case of a stacked ensemble). The performance of Naïve Bayes classifier drops significantly using chi-square test after selecting ~3.125% of original features, whereas for other selection methods the convergence towards a weak baseline is slower. This collection of features also ensured the best performance for Naïve Bayes classifier. The optimal model for bigram representation was obtained by considering all of the features and using a stacked ensemble classifier. However, approximately the same results were obtained by using logistic regression on 6.25% of original features. The optimal performance for each classifier and weighting scheme are presented in Table 8.

Table 8. Optimal bigram model performance by weighting method.

		F1	**AUC**	**Feature selection metric**	**% / number of features chosen**
Naïve Bayes	TF	0.615	0.754	Chi-square	1.562% / ~135
	TFC	0.656	0.789	Information gain	3.125% / ~265
Stacked ensemble	TF	0.708	0.895	Chi-square	6.25% / ~530
	TFC	**0.75**	**0.926**	None	100% / ~8500
XGBoost	TF	0.69	0.88	Information gain	6.25% / ~530
	TFC	0.718	0.908	Information gain	
Logistic regression	TF	0.70	0.893	Information gain	6.25% / ~530
	TFC	0.746	0.922	None	100% / ~8500

It is evident that the performance of stacked ensembling on bigrams is equivalent to the one obtained on unigrams, yet the performance of other algorithms tends to deteriorate. Besides this, having an appropriate weighting scheme is presumably relevant for bigrams, as each algorithm seems to perform significantly better using TFC weighting compared to TF. In view of the fact that using ~8500 bigrams achieves the same result as using ~375 unigrams, the use of bigrams remains unjustified and will not be analysed more thoroughly.

Trigrams

The number of trigrams is appreciably smaller (even less than there are unigrams). The results on Figure 5 are reported using TFC weighting and the outcome of TF weighting is presented in Appendix 3. In case of trigrams, the complete data baseline in terms of F1-score was slightly below 0.7 for logistic regression and stacked ensembling. On the other hand, XGBoost and Naïve Bayes perform noticeably worse – their best performance is correspondingly 0.676 and 0.539. Once again, the difference of F1-scores for two best models is insignificant.

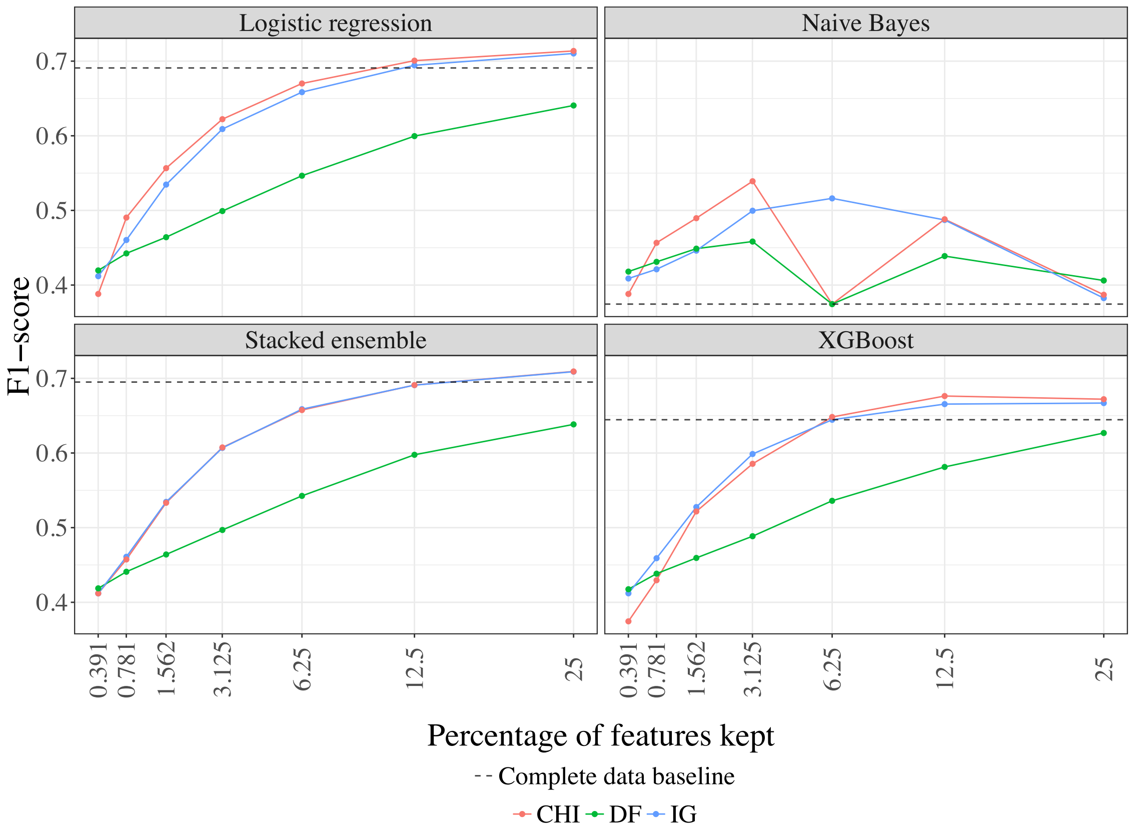


Figure 5. Trigram classifier performance by feature selection method using TFC weighting.

It can be seen that an appropriate weighting scheme for trigrams is especially important, as the performances differ noticeably. It is evident that Naïve Bayes struggles to learn anything useful using TF weighting. For other models, the complete data baseline is slightly worse than the optimal results that were obtained by considering a subset of features. The optimal performance was achieved by selecting ~25% of features (~625 trigrams) using any of the supervised feature selection methods. In case of document frequency thresholding, the performances generally did not converge near the optimal value as they did before (except when using TF weighting and XGBoost). The optimal performances for every model with corresponding weighting method are summarised in Table 9.

It is evident that the overall best result in terms of F1-score was obtained using logistic regression – the precision and recall were correspondingly 0.68 and 0.76. However, in terms of AUC value, a slightly better result was obtained using stacked ensembling. In this case, the precision and recall were correspondingly 0.67 and 0.755.

Table 9. Optimal trigram model performance by weighting method.

		F1	**AUC**	**Feature selection metric**	**% / number of features chosen**
Naïve Bayes	TF	0.4	0.54	DF thresholding	1.562% / ~40
	TFC	0.539	0.693	Chi-square	3.125% / ~80
Stacked ensemble	TF	0.7	0.89	Information gain	50% / ~1250
	TFC	0.709	**0.907**	Chi quare	25% / ~625
XGBoost	TF	0.59	0.795	Chi-square	25% / ~625
	TFC	0.676	0.86	Chi-square	12.5% / ~315
Logistic regression	TF	0.7	0.89	Information gain	50% / ~1250
	TFC	**0.713**	0.904	Chi-square	25% / ~625

For a trigram representation, it is preferable to employ logistic regression compared to stacked ensembling due to the slightly better precision and recall values. However, as none of the above-presented combinations outperform the simple unigram model, then the author will not go into any further details of given models.

Bag of uni-, bi- and trigrams

As every representation method seems to have its own strengths, then for the following experiments, the author combined them all together. The results on Figure 6 are reported using TFC weighting and the same progress on TF weighting is presented in Appendix 4.

It is evident that document frequency thresholding ensures noticeably worse results for every model except stacked ensembling. However, the results are noticeably better using TF weighting – document frequency thresholding seems to perform comparably well for every model except Naïve Bayes. Furthermore, for Naïve Bayes the latter seems to ensure appreciably better results when using TF weighting. The fastest convergence towards a complete data baselines was ensured using XGBoost. However, its optimal result is outperformed by logistic regression and stacked ensemble. The best result for every model except Naïve Bayes was obtained using ~6.25% of features (~875). On the other hand, the optimal Naïve Bayes model was achieved using 1.562% (~220) features. However, as it is substantially worse than other methods, then it will not be considered.

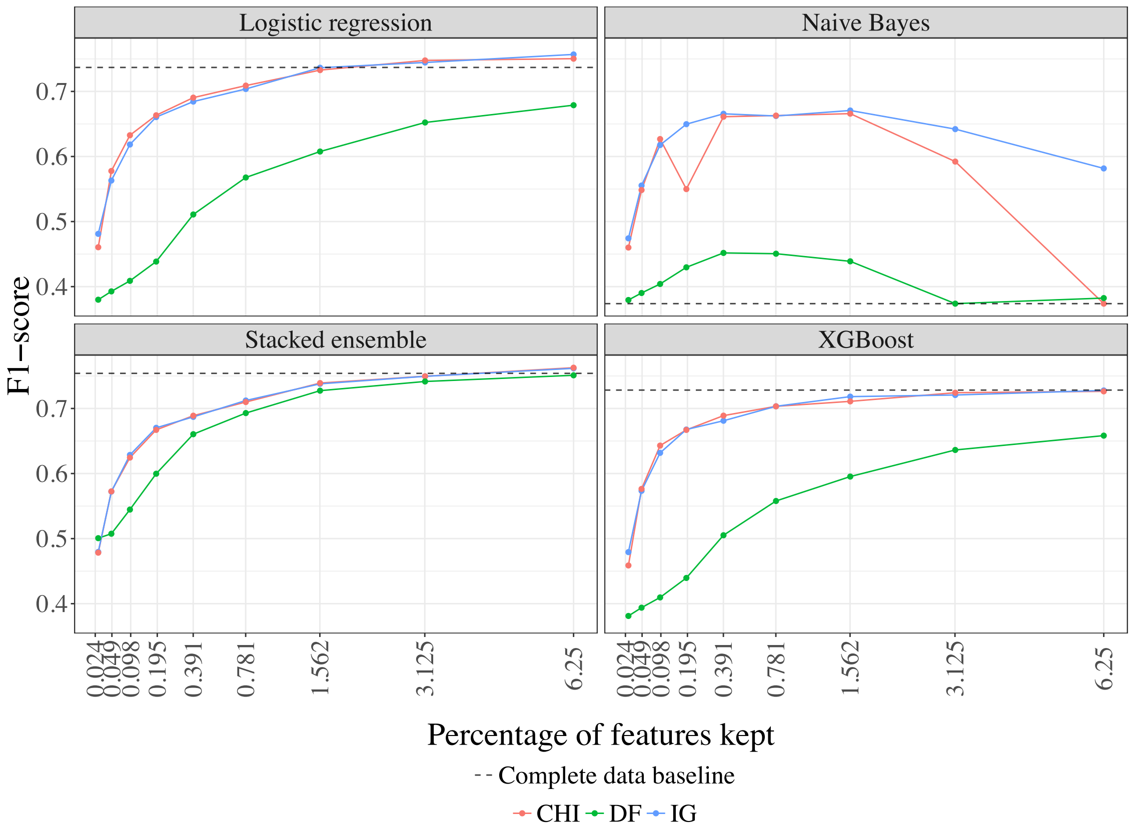


Figure 6. Uni-, bi- and trigram classifier performances by feature selection method using TFC weighting.

Document frequency thresholding ensures nearly an optimal result for stacked ensemble, whereas for other algorithms the F1-score does not seem to converge near the optimum. The optimal model performances are presented in Table 10.

Table 10. Uni-, bi- and trigram optimal model performances by weighting method.

		F1	**AUC**	**Feature selection metric**	**% / number of features chosen**
Naïve Bayes	TF	0.682	0.81	Information gain	1.562% / ~220
	TFC	0.67	0.8		
Stacked ensemble	TF	0.757	0.924	None	100% / ~14000
	TFC	**0.762**	0.931	Chi-square	6.25% / ~875
XGBoost	TF	0.74	0.918	None	100% / ~14000
	TFC	0.727	0.916	Information gain	6.25% / ~875
Logistic regression	TF	0.751	0.924	Information gain	6.25% / ~875
	TFC	0.756	**0.933**		

It can be seen that for Naïve Bayes and XGBoost, the optimal performance was achieved using TF weighting, whereas for other algorithms, using TFC ensured a slightly better result. The best performance for XGBoost was achieved by using all ~14000 features, yet it is not advised due to the computational complexity. The optimal model in terms of F1-score was achieved by using ~875 features and a stacked ensemble. In this case, the precision and recall were correspondingly 0.716 and 0.816. However, nearly the same result was obtained using the same number of features with logistic regression, where the precision and recall were correspondingly 0.723 and 0.798.

Two-skip-bigrams

By far the computationally heaviest experiments were conducted using two-skip-bigrams. The results obtained on TF weighting are presented in Appendix 5. The complete data baseline for latter was established only for logistic regression, as other methods were not capable of handling ~25000 features. The results on Figure 7 are reported using TFC weighting.

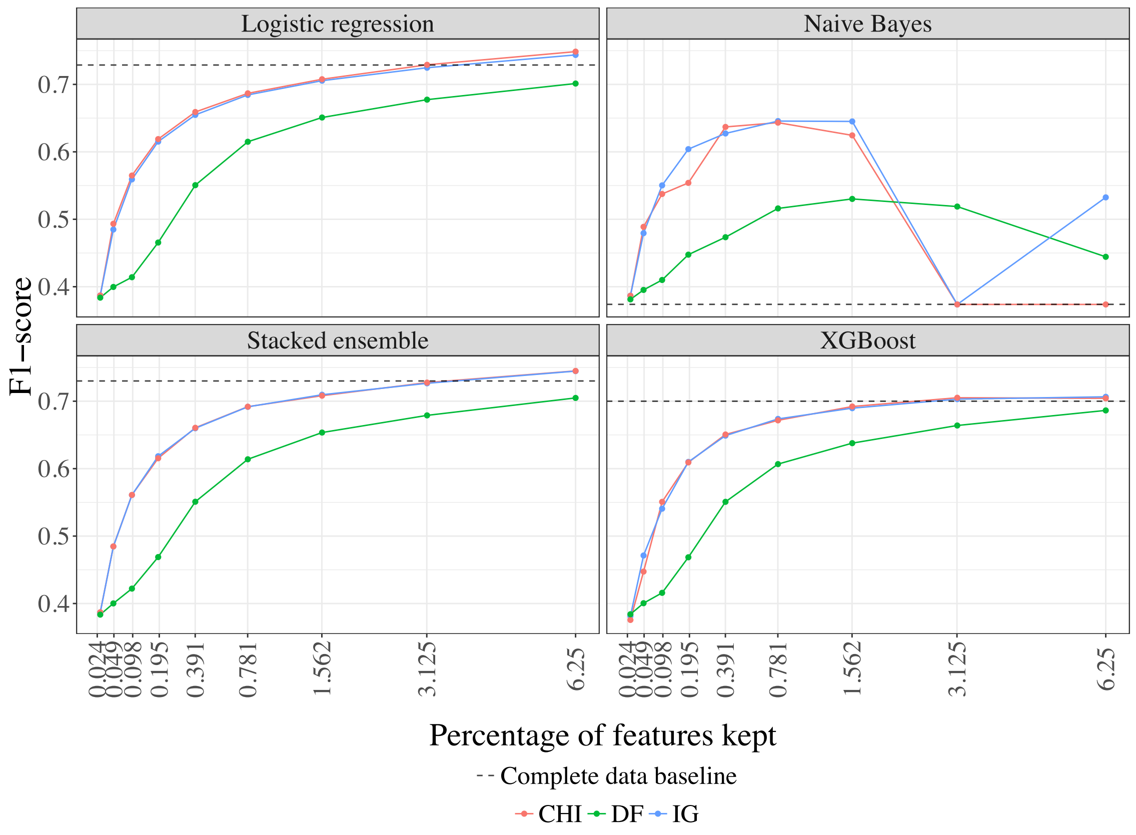


Figure 7. Two-skip-bigram classifier performances by feature selection method using TFC weighting.

Once again it is evident, that logistic regression and stacked ensemble ensure similar performances. On the other hand, the optimal performance of XGBoost is slightly worse compared to the aforementioned models. It can be seen, that for large dimensional datasets, information gain and document frequency thresholding ensure indistinguishably similar results. For every model except Naïve Bayes, the convergence towards complete data baseline was achieved using at least 3.125% (~785) features. Considering the computational complexity added by including additional features, the author sees using only ~1% of features sufficient, as the incremental improvements after this are marginal. Furthermore, the author observed that considering more than 6.25% of features do not enhance the performance any further. The optimal performances for every model are presented in Table 11.

Table 11. Two-skip-bigram optimal model performances by weighting method.

		F1	**AUC**	**Feature selection metric**	**% / number of features chosen**
Naïve Bayes	TF	0.652	0.78	Information gain	0.781% / ~195
	TFC	0.645	0.773		
Stacked ensemble	TF	0.735	0.919	Information gain	6.25% / ~1565
	TFC	0.745	**0.926**	Chi-square	
XGBoost	TF	0.715	0.9	Information gain	6.25% / ~1565
	TFC	0.706	0.897		
Logistic regression	TF	0.73	0.919	Information gain	6.25% / ~1565
	TFC	**0.748**	0.923	Chi-square	

It can be seen that the choice of weighting scheme does not affect the performance remarkably. For XGBoost and Naïve Bayes, the optimal performance was achieved using TF weighting, yet the difference compared to TFC weighting is marginal. The best model in terms of F1-score was achieved using ~1565 features deemed important by chi-square statistic. The F1-score of 0.748 was obtained by recall and precision values being correspondingly 0.77 and 0.736. On the other hand, the best model in terms of AUC value (0.926) was obtained using a stacked ensemble, which had recall and precision values correspondingly equal to 0.72 and 0.77. Considering the interpretability and performances of both, the author sees logistic regression as a better choice.

Latent semantic analysis

To ascertain if methods that aim to capture the semantic meaning of words further enhance the performance, the author employed latent semantic analysis using TF and TFC weighting. On Figure 8, the number of factors was varied proportionally to the fraction of words that were considered in the unigram feature selection process.

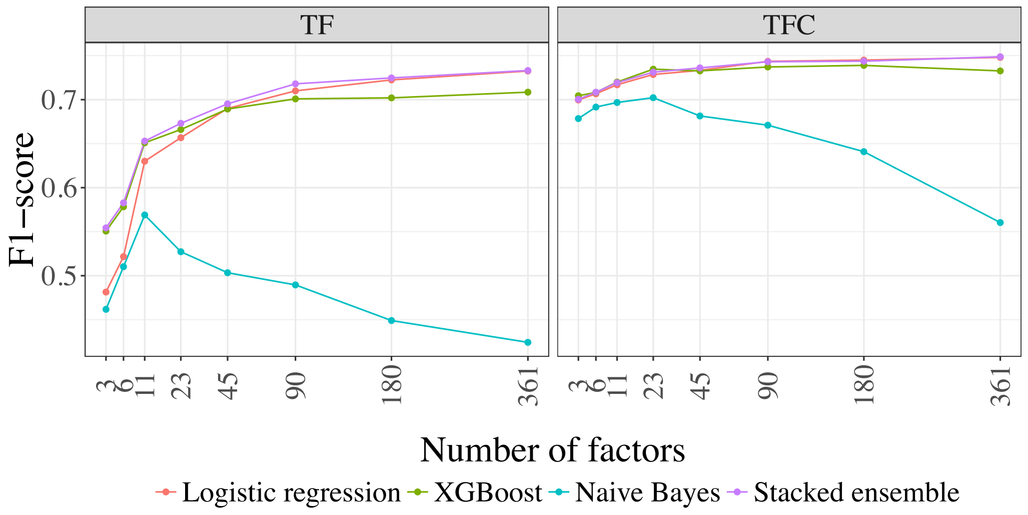


Figure 8. LSA classification performance by weighting method and number of factors.

As it can be seen, then an appropriate weighting scheme is relevant for latent semantic analysis – the optimal performance is nearly achieved by using only 3 factors and TFC weighting. However, considering additional factors does not seem to improve the performance remarkably in this case. The optimal performances using latent semantic analysis are presented in Table 12.

Table 12. LSA optimal model performances by weighting method.

		F1	AUC	Number of factors
Naïve Bayes	TF	0.57	0.79	11
	TFC	0.7	0.89	23
Stacked ensemble	TF	0.733	0.91	361
	TFC	0.743	0.922	
XGBoost	TF	0.709	0.9	361
	TFC	0.738	0.918	
Logistic regression	TF	0.732	0.909	361
	TFC	**0.745**	**0.923**	

In given case, the models tend to converge to similar performances regardless of employed weighting method. Although using latent semantic analysis does not generally provide a better result, then for Naïve Bayes it ensured the best performance compared to every other representation method. The optimal performance is obtained by using logistic regression, 361 factors and TFC weighting – the corresponding F1 and AUC values are

0.745 and 0.923. In that case, the precision and recall were correspondingly 0.72 and 0.774. However, as the factors are not directly interpretable and using LSA does not outperform the current benchmark, then it is not favored. Furthermore, the performance is actually worse than of any other classical text representation method (except trigrams).

The optimal interpretable and complex models for every classical representation method are summarised in Table 13.

Table 13. Optimal models for classical representation methods.

Rank	Algorithm	Representation	F1	AUC	# of predictors
1.	Stacked ensemble	Uni-, bi- and trigrams	0.762	0.931	~875
	Logistic regression		0.756	0.933	
2./3.	Stacked ensemble	Unigrams	0.75	0.925	~375
	Logistic regression		0.747	0.927	
2./3.	Stacked ensemble	Bigrams	0.75	0.926	~8500
	Logistic regression		0.746	0.922	
4.	Stacked ensemble	Two-skip-bigrams	0.745	0.926	~1565
	Logistic regression		0.748	0.923	
5.	Stacked ensemble	LSA	0.743	0.91	361
	Logistic regression		0.745	0.923	
6.	Stacked ensemble	Trigrams	0.709	0.907	~625
	Logistic regression		0.713	0.904	

It is evident that models built upon any representation except trigrams perform alike. Combining uni-, bi- and trigrams ensures a slightly better performance than other methods. It appeared that for every representation, the optimal complex model was stacked ensemble and the optimal interpretable model was logistic regression. Harnessing bigrams and two-skip-bigrams is not advised due to the large dimensionality and the training complexity accompanied by this. Furthermore, it is evident that optimal complex models do not outperform logistic regression noticeably. Referring to the analysis provided above, the author favours using logistic regression and a combination of uni-, bi- and trigrams. Concisely, this is the established baseline that will be considered for further analysis.

5.2 Novel methods benchmark

To evaluate the expediency of recognising the semantic meaning of words, the author employed *word2vec* embeddings by creating a bag of concepts representation such that the number of clusters was varied proportionally to the fraction of words that were considered in the unigram feature selection process. As a baseline, the author used simple coordinate-wise averaging such that all word embeddings in a specific document were summed up and the resulting vector was divided by the number of words in given document. The progress of obtaining an optimal model using *word2vec* embeddings is presented on Figure 9.

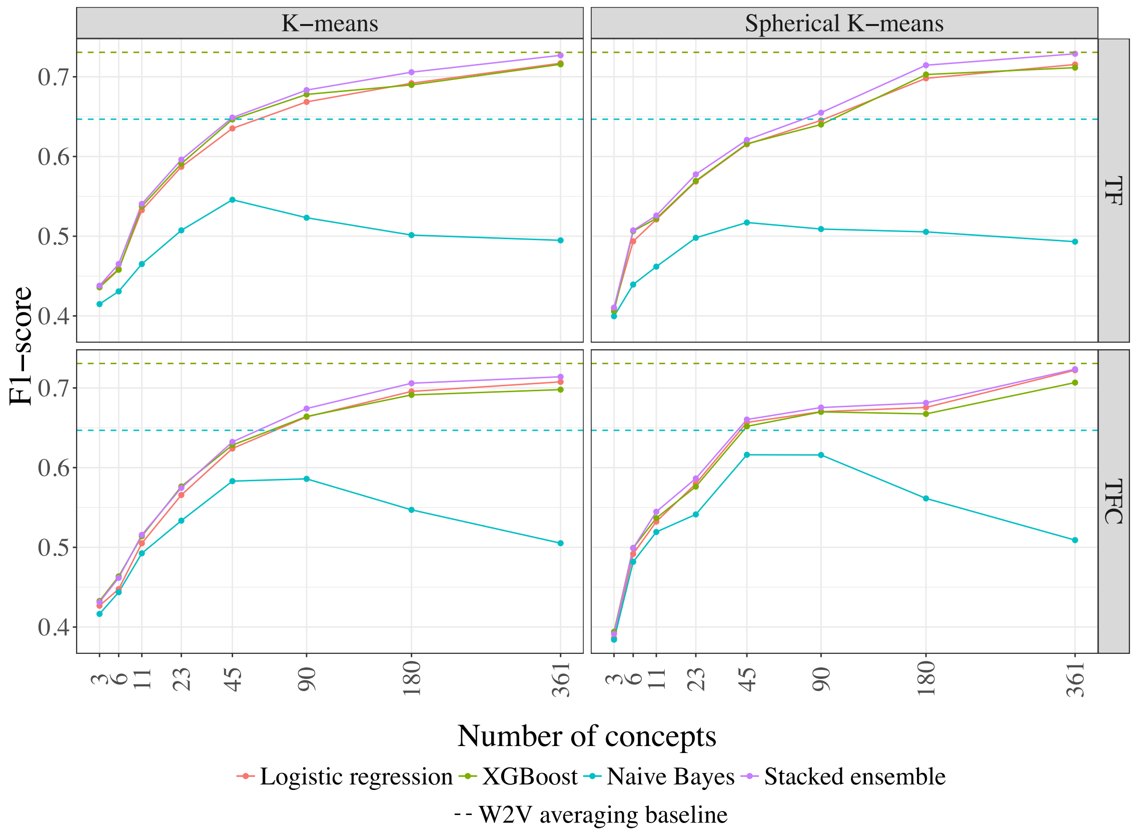


Figure 9. Bag of concepts performance by weighting method and number of concepts.

It is evident that the weighting scheme, as well as the distance measure in given context have no noticeable impact on the performance. Furthermore, despite the choice of underlying method, the best performance on bag of concepts representation was obtained using a stacked ensemble. However, in most of the cases logistic regression behaves almost indistinguishably similarly and is advised due to its interpretability. Naïve Bayes on the other hand performs noticeably worse than other algorithms and it is the only algorithm that does not converge to the baseline obtained by averaging the word embeddings. It can be seen that simply averaging the embeddings provides a relatively strong baseline, yet from a practical perspective using a bag of concepts is preferred due to its interpretability. To have a better overview of the concepts, the author performed a sanity check and listed random subsets of words belonging to clusters formed by both clustering methods in Table 14. It needs to be stated, that the concepts are not ordered, thus they do not need to be similar for both methods.

Table 14. An example of grouped terms by clustering method.

	K-means		
	Concept 1	**Concept 2**	**Concept 3**
Words	happy pleased impressed satisfied excited	inconvenience delay glitch delayed	dollar euro rand rupee peso
	Spherical K-means		
	Concept 1	**Concept 2**	**Concept 3**
Words	companies businesses providers vendors suppliers	indicate reflect indication suggest indicating	dollar currency euro rand sterling

It can be seen that both clustering methods work relatively well, as the concepts are meaningful in given context. For k-means, concepts 1 and 2 both indicate customer satisfaction, although different polarity, thus grouping them differently seems reasonable. Same applies for currency-related concepts – both methods happened to have one such cluster. For spherical k-means, concept 1 could be labelled as organisations and concept two seems to resemble any indications. It is evident that both clustering methods perform relatively well, as the clusters seem reasonable.

Concisely, the optimal results for stacked ensembling and logistic regression are alike, yet considering their interpretability, using the latter is advisable. The benchmark for novel methods was obtained using logistic regression, TFC weighting and spherical clustering with 361 clusters. Its F1-score of 0.72 is ensured by precision and recall values being correspondingly 0.68 and 0.77. Although it is not better than the optimal model obtained on a classical representation, it needs to be stated that both representations have their own strengths, which may make them qualitatively exceedingly different. Therefore it is beneficial to evaluate its suitability in the next subchapter.

5.3 Optimal model comparison

Subsequently, the author will compare the performances of optimal models obtained on both classical (logistic regression with 6.25% of uni-, bi- and trigrams) and novel (logistic regression with 361 spherical clusters) representations. The purpose of this subchapter is to evaluate the practical applicability of given models quantitatively as well as qualitatively. The performance of comparable models is recalled in Table 15.

Table 15. Optimal novel and classical method performances.

	F1	**AUC**	**Precision**	**Recall**
Novel benchmark	0.722	0.91	0.685	0.768
Classical benchmark	0.756	0.933	0.723	0.797

The optimal model on classical representation was achieved using the following parameters – $\alpha = 0$ & $\lambda = 0.0332$. This means that the weights of the model were penalised using only Ridge regression. The latter indicates that the feature selection process was efficient, as the optimal result was obtained using all the predictors. Out of ~875 features used, 36% were unigrams, 48% bigrams and 16% trigrams. On the other hand, the novel benchmark was obtained using elastic net penalty, such that $\alpha = 0.2$ & $\lambda = 0.0536$. This indicates, that some learned concepts were irrelevant and using a feature selection process for given representation may be beneficial.

To provide an overview of the qualitative differences between aforementioned models, the author will disclose and rank the 5 most important terms in Table 16 based on the learned weights. The author will also provide if they increase (+) or decrease (–) the probability of the contact being related to the problem of our interest. Furthermore, for the bag of concepts benchmark, the author will provide two terms related to the corresponding concepts. From a qualitative perspective it is evident, that both models use remarkably different features. This indicates the potential of stacking, as the models seemingly cover a different prediction space.

Table 16. TOP5 predictors for classical and novel benchmark models.

Classical benchmark	Novel benchmark	
1. status (+)	1. concept 96 (+)	(question, query)
2. transfer (+)	2. concept 115 (–)	(launched, introduced)
3. status transfer (+)	3. concept 251 (+)	(permanently, temporarily)
4. money (+)	4. concept 346 (+)	(difficult, challenging)
5. card (–)	5. concept 276 (–)	(property, land)

To have a better overview of the covered prediction space, the author has visualised the relevant class predictions (only the ones we are interested in, the minority class) that were the same for both models on Figure 10. Apropos, the color indicates if the decision was correct.

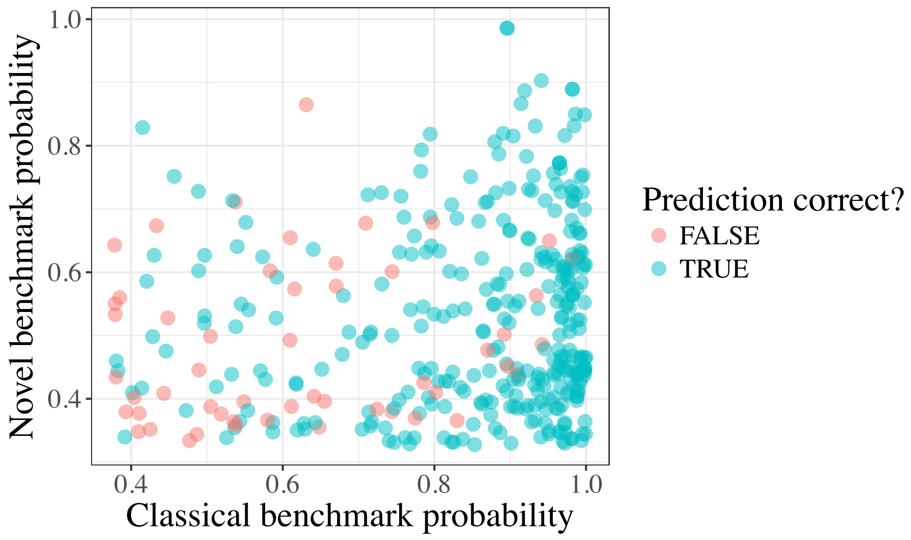


Figure 10. Relevant class predictions and their correctness.

It is evident that the models make distinguishably different mistakes, as the union of their predictions only covers ~10% of all relevant cases. However, both models are good on their own. Furthermore, it can be seen that for cases where the classical benchmark is very certain in its prediction (probability near 1), the probability assigned by the novel benchmark varies greatly. As the predictions are rather diverse, then it confirms the potential of stacking, such that the base learners are obtained on different (classical and novel) representation schemes.

To apply any of these benchmark models in practice, we need to be sure they generalise on unseen data, thus we have not overfitted them. To evaluate their suitability, the author has visualised corresponding ROC curves obtained on training and cross-validation sets on Figure 11.

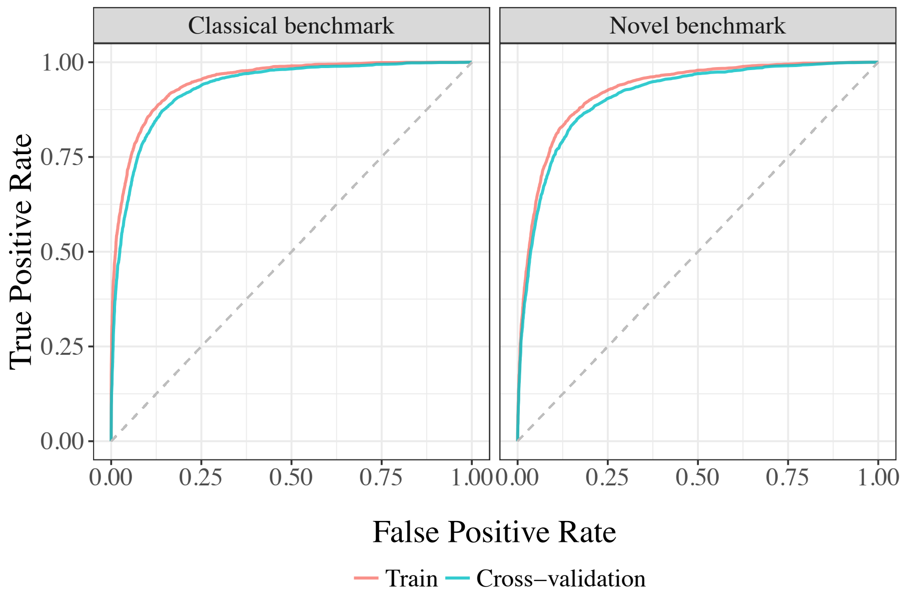


Figure 11. Train and cross-validation sets ROC curves.

It can be seen that there is no evident indication of overfitting – the curves reported on train and cross-validation sets are relatively similar. Therefore we can be certain, that the models apply in practice as expected.

To have a more principled overview of their applicability, the author will visualise corresponding cross-validation ROC curves on Figure 12.

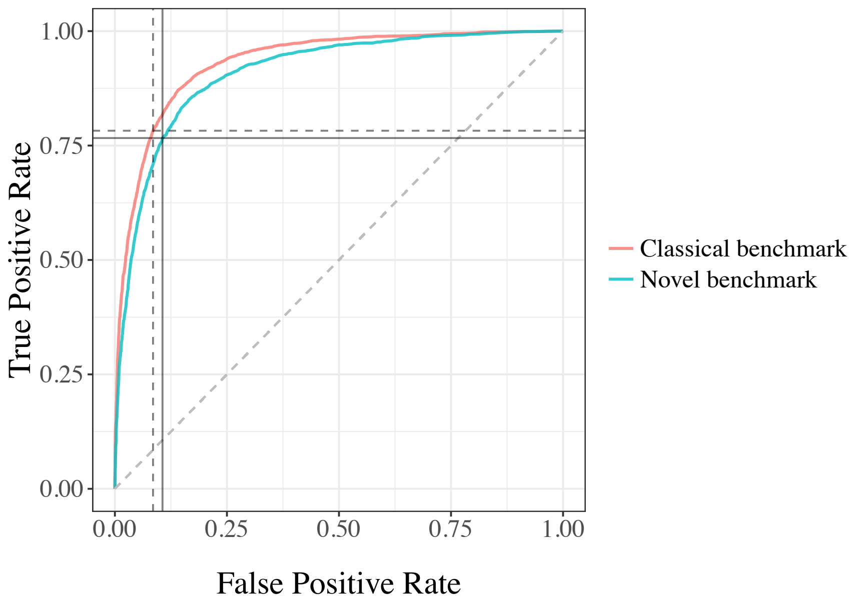


Figure 12. ROC curves of benchmark models.

The author has also noted the decision thresholds (horisontal and vertical lines) that ensure an acceptable level of false and true positives that is necessary for applying the models in practice. For the classical benchmark, the decision threshold which maximises the F1-score ensures false and true positive rates correspondingly 0.0851 and 0.7824. In other words, the model is able to detect ~78.2% of relevant cases such that ~8.5% of contacts that are not of our interest are falsely marked as relevant. For the novel approach, the corresponding values are ~76.6% and 10.6%.

Considering the above-presented analysis, the author sees both models applicable in practice, yet using the one obtained on classical representation is favoured.

Conclusion

In a world of stiff competition, it is crucial for companies to build decision support systems that can extract useful information from one of the most abundant sources of information – written text. Given thesis was with a practical nature and the purpose of it was to implement an automated text analysis model using data from TransferWise Ltd. that can be used to efficiently prioritise and measure incoming customer contacts. The practical application of given thesis is particularly important – automatically analysing thousands of customer inquiries provides exceptionally valuable insights that can be used to optimise operational procedures and solve customer problems in the most convenient way.

In the first part of the thesis, the author gave an overview of the practical motivation behind building automated text analysis solutions. Thereby the author also affirmed the potential of other apparent text mining opportunities. The theoretical part mainly focused on providing a thorough overview of applicable text representation, feature engineering and machine learning methods. Apropos, besides prevalent methods, the author also considered novel text representation techniques like *word2vec* and complex classification algorithms like *xgboost* and stacking. The empirical part relied on a corpus compiled by the author using data from TransferWise Ltd. To establish the optimal model, the author carried out numerous experiments on every method combination by using grid search and stratified 10-fold cross-validation. The numerous models were ranked based on their F1-score and the suitability of top performing models was respectively analysed using AUC value and ROC curve analysis as well as the building blocks of F1-score – the precision and recall.

The tangible model was established using logistic regression with a mixture of uni-, bi- and trigrams. It was sufficient to only consider 6.25% of most relevant terms based on either information gain or chi-square statistic. It appeared that both supervised feature selection methods broadly ensured alike performances. Furthermore, the only unsupervised feature selection criteria, document frequency thresholding, generally converged to the same optimum, but at a slower pace. It was evident, that for rather short texts like emails, the choice of weighting scheme is mostly irrelevant. Similarly, context-aware embeddings like *word2vec* and latent semantic analysis did not ensure a better result. In case of underlying classification algorithms, it was evident that the superiority of more complex models is insignificant and considering their lack of interpretability, the use of them remains unjustified.

The purpose of given thesis was fulfilled – the established model was able to detect on average ~80% of contacts related to the problem of our interest such that ~72% of its relevant class predictions were correct. Although there is room for further enhancements, this is a relatively strong baseline and the author sees the model efficiently applicable in practice. The model is considerably better than a first-in-first-out approach, thus it can be used for prioritising incoming contacts.

As a next step, the author sees retraining the model on a significantly larger dataset and integrating it to operational procedures. Besides this, the author emphasizes the potential of complexity based prioritisation, thus the similar framework shall be applied on a slightly different assignment. From a theoretical perspective, the author sees the potential of employing alternative distance measures for compiling a bag of concepts representation as well as compiling the same representation on sequences of words instead of unigrams. Furthermore, as models trained on novel and classical representations made distinguishably different mistakes, then it can be beneficial to train a stacked ensemble by using corresponding representations for different base learners.

Bibliography

1. **Aggarwal, C. C., Zhai, C. X.** A Survey of Text Classification Algorithms. – Mining Text Data, Springer, 2012, pp. 163-222, URL: https://pdfs.semanticscholar.org/-5c89/852b90a1e9e506d237749c745bf42ac0f737.pdf

2. **Apte, C., Damerau, F., Weiss, M., S.** Automated Learning of Decision Rules for Text Categorization. – ACM Transactions on Information Systems, Vol. 12, No. 3, 1994, pp. 233-251.

3. **Arnott, D., Lizama, F., Song, Y.** Patterns of Business Intelligence Systems Use in Organisations. – Decision Support Systems, 2017, Vol. 97, pp. 58-68, DOI: 10.1016/-j.dss.2017.03.005

4. **Bernotas, M., Karklius, K., Laurutis, R., Slotkiene, A.** The Peculiarities of the Text Document Representation Using Ontology and Tagging-Based Clustering Technique. – Information Technology and Control, Vol. 36, No. 2, 2007, pp. 217-220, URL: http://www.itc.ktu.lt/index.php/ITC/article/viewFile/11834/6500

5. **Breiman, L.** Random Forests. – Machine Learning, 2001, Vol 45, pp. 5–32, URL: https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf

6. **Chen, T., Guestrin, C.** XGBoost: A Scalable Tree Boosting System. – Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785-794, DOI: 10.1145/2939672.2939785

7. **D'hondt, E., Verberne, S., Weber, N., Koster, K., Boves, L.** Using Skip-grams and PoS-based Feature Selection for Patent Classification. – Computational Linguistics in the Netherlands Journal, Vol. 2, 2012, pp. 52-70, URL: http://repository.ubn.ru.nl/bitstream-/handle/2066/102120/102120.pdf

8. **Da Silva, N. F. F., Hruschka, R. E., Hruschka R. E. Jr.** Tweet Sentiment Analysis With Classifier Ensembles. – Decision Support Systems, Vol. 66, 2014, pp. 170-179, DOI: 10.1016/j.dss.2014.07.003

9. Deerwester, S., Dumais, T. S., Furnas, W. G., Landauer, K. T., Harshman, R. Indexing by Latent Semantic Analysis. – Journal of the American Society for Information Science, Vol. 41, No. 6, 1990, pp. 391-407, URL: http://www.psychology.-uwo.ca/faculty/harshman/latentsa.pdf

10. Dhillon, S. I., Modha, S. D. Concept Decompositions for Large Sparse Text Data using Clustering. – Machine Learning, 2001, Vol. 42, No. 1-2, pp. 143-175, DOI: 10.1023/A:1007612920971

11. Dörre, J., Gerstl P., Seiffert, R. Text Mining: Finding Nuggets in Mountains of Textual Data. – Proceedings of The Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1999, pp. 398-401, URL: http://cs.uvm.edu/~xwu/kdd/-KDD-dorre.pdf

12. Fawcett, T. ROC Graphs: Notes and Practical Considerations for Data Mining Researchers, 2003, Technical Report, HP Laboratories, pp. 1–28, URL: http://www.hpl.hp.com/techreports/2003/HPL-2003-4.pdf

13. Forman, G. An Extensive Empirical Study of Feature Selection Metrics for Text Classification. – Journal of Machine Learning Research, 2003, pp. 1289-1305, URL: http://www.jmlr.org/papers/volume3/forman03a/forman03a.pdf

14. Friedman, H. J. Stochastic Gradient Boosting. – Computational Statistics & Data Analysis, 2002, Vol. 38, No. 4, pp. 367-378, DOI: 10.1016/S0167-9473(01)00065-2

15. Fürnkranz, J. A Study Using n-gram Features for Text Categorization. – Austrian Research Institute for Artificial Intelligence, 1998, pp. 1-10, URL: https://pdfs.semanticscholar.org/3a4f/7eca26ee30aeb798a0eef2c08dd7a1cbad01.pdf

16. Gao, L., Chang, E., Han, S. Powerful Tool to Expand Business Intelligence: Text Mining. – International Journal of Computer and Information Engineering, 2007, Vol. 1, No. 8, pp. 110-115, URL: https://espace.curtin.edu.au/handle/20.500.11937/17233

17. Golberg, Y., Levy, O. Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method. – arXiv Preprint, 2014, 5p, URL: arxiv.org/pdf/-1402.3722.pdf

18. Google Code Archive. – https://code.google.com/archive/p/word2vec/ (accessed on 04.18.2018)

19. Gu, Q., Cai, Z., Zhu, L., Huang, B. Data Mining on Imbalanced Datasets. – International Conference on Advanced Computer Theory and Engineering, 2008, pp. 1020-1024, DOI: 10.1109/ICACTE.2008.26

20. Gupta, V., Lehal, S. G. A Surve of Text Mining Techniques and Applications. – Journal of Emerging Technologies in Web Intelligence, 2009, Vol. 1, No. 1, pp. 60-76, URL: http://www.jetwi.us/uploadfile/2014/1230/20141230112729939.pdf

21. Guthrie, D., Allison, B., Liu, W., Guthrie, L., Wilks, Y. A Closer Look at Skip-gram Modelling. – Proceedings of the 5th international Conference on Language Resources and Evaluation, 2006, pp. 1222-1225, URL: http://www.cs.brandeis.edu-/~marc/misc/proceedings/lrec-2006/pdf/357_pdf.pdf

22. Hastie, R., Tibshirani, R., Friedman, J. The Elements of Statistical Learning: Data Mining, Inference and Prediction, Second Edition, 2009, 745p.

23. Huang, A. Similarity Measures for Text Document Clustering. – The Sixth New Zealand Computer Science Research Student Conference, 2008, pp. 49-56, URL: https://goo.gl/HzPtnH

24. Ikonomakis, M., Kotsiantis, S., Tampakas, V. Text Classification Using Machine Learning Techniques. – WSEAS Transactions on Computers, Issue 8, Vol. 4, 2005, pp. 966-974, URL: citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.9153&rep=-rep1&type=pdf

25. Irwin, M. S. A., Choo, R. K., Liu, L. An analysis of money laundering and terrorism financing typologies. – Journal of Money Laundering Control, 2011, Vol 15, No. 1, pp. 85–111, DOI: http://dx.doi.org/10.1108/13685201211194745

26. Jurafsky, D., Martin, H. J. Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition, Third Edition draft, 2017, 499 p.

27. Jurgovsky, J., Granitzer, M., Seifert, C. Evaluating Memory Efficiency and Robustness of Word Embeddings. – Advances in Information Retrieval, 2016, pp. 200-211, DOI: 10.1007/978-3-319-30671-1_15

28. Khoo, A., Marom, Y., Albrecht, D. Experiments with Sentence Classification. – Proceedings of the 2006 Australasian Language Technology Workshop, 2006, pp. 18-25, URL: http://www.aclweb.org/anthology/U06-1005

29. Kim, H., K., Kim, H., Cho, S. Bag-of-Concepts: Comprehending document representation through clustering words in distributed representation. – Neurocomputing, Vol. 266, 2017, pp. 336-352, DOI: 10.1016/j.neucom.2017.05.046.

30. Kohavi, R. A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection. – International Joint Conference on Articial Intelligence, Vol. 14, No. 2, 1995, pp. 1137-1145, URL: https://pdfs.semanticscholar.org/0be0/d781305750b37-acb35fa187febd8db67bfcc.pdf

31. Kohavi, R., John, G. H. Wrappers for feature subset selection. – Artificial Intelligence, 1997, pp. 273-324, DOI: 10.1016/S0004-3702(97)00043-X

32. Kuhn, M., Johnson K. Applied Predictive Modeling. – Springer Science + Business Media, New York, 574 p.

33. Laan, V., Mark, J., Polley, E. C., Hubbard, A. E. Super Learner. -– Statistical applications in genetics and molecular biology, 2007, Vol. 6, No. 1, 21p, URL: https://pdfs.semanticscholar.org/19e9/c732082706f39d2ba12845851309714db135.pdf

34. Lewis, D., D. An evaluation of phrasal and clustered representations on a text categorization task. – Proceedings of the 15th annual international ACM SIGIR conference on Research and Development in Information Retrieval, 1992, pp. 37-50, URL:citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.81.8161&rep=rep1&type=pdf

35. Lilleberg, J., Zhu, Y., Zhang, Y. Support Vector Machines and Word2vec for Text Classification with Semantic Features. – Cognitive Informatics & Cognitive Computing, 2015, pp. 136-140, DOI: 10.1109/ICCI-CC.2015.7259377.

36. Liu, Y., Gu, Y., Nguyen, C. J., Li, H., Zhang, J., Gao, Y., Huang, Y. Symptom Severity Classification with Gradient Tree Boosting. – Journal of Biomedical Informatics, 2017, Vol. 75 Supplement, pp. 105-111, DOI: 10.1016/j.jbi.2017.05.015

37. Mar, W. K., Naing, T. T. Optimum Neural Network Architecture for Precipitation Prediction of Myanmar. – International Journal of Environmental, Chemical, Ecological, Geological and Geophysical Engineering, 2008, Vol. 2, No. 12, pp. 154–158, URL: http://www.waset.org/publications/13861

38. Mikolov, T., Deoras, A., Kombrink, S., Burget, L., Cernocky, J. Empirical Evaluation and Combination of Advanced Language Modeling Techniques. – 12th Annual Conference of the International Speech Communication Association, 2011, pp. 605-608, URL: https://www.isca-speech.org/archive/archive_papers/inter-speech_2011/i11_0605.pdf

39. Mikolov, T., Le, Q. Distributed representations of sentences and documents. – International Conference on Machine Learning, 2014, pp. 1188-1196, URL: http://proceedings.mlr.press/v32/le14.pdf

40. Murphy, P. K. Machine Learning – A Probabilistic Perspective, 2012, 1098p.

41. Passalis, N. Tefas, A. Neural Bag-of-Features Learning. – Pattern Recognition, 2017, Vol. 64, pp. 277-294, DOI: 10.1016/j.patcog.2016.11.014

42. Peng, F., Schuurmans, D., Wang, S. Augmenting Naïve Bayes Classifiers with Statistical Language Models. –– Information Retrieval, 2004, Vol. 7, No. 3-4, pp. 317--345, URL: https://scholarworks.umass.edu/cgi/viewcontent.cgi?article=1090&context-=cs_faculty_pubs

43. Sahlgren, M., Cöster, R. Using Bag-of-Concepts to Improve the Performance of Support Vector Machines in Text Categorization. – Proceedings of the 20th international conference on Computational Linguistics, Article No. 487, 2004, 5 p, DOI: 10.3115/1220355.1220425

44. Salton G., Buckley, C. Term-weighting approaches in automatic text retrieval. – Information Processing and Management, Vol. 24, No. 5, 1988, pp. 513-523, DOI: 10.1016/0306-4573(88)90021-0

45. Schütze, H., Hull, A. D., Pedersen, O. J. A Comparison of Classifiers and Document Representations for the Routing Problem. – Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1995, 9 p, URL: http://lsa3.colorado.edu/LexicalSemantics/-schutze95comparison.pdf

46. Sebastiani, F. Machine Learning in Automated Text Categorization. – ACM computing surveys (CSUR), Vol. 34, Issue. 1, 2002, pp. 1-47, DOI: 10.1145/-505282.505283

47. Silva, C., Ribeiro, B. The importance of stop word removal on recall values in text categorization. – Proceedings of the International Joint Conference on Neural Networks, 2003, pp. 1661-1666, DOI: 10.1109/IJCNN.2003.1223656

48. Slonim, M., Tishby, N. The Power of Word Clusters for Text Classification. – 23rd European Colloquium on Information Retrieval Research, Vol. 1, 2001, pp. 1-12, URL: citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.22.7039&rep=rep1&type=pdf

49. Wang, Y. F., Tan, C. M., Lee, C. D. The Use of Bigrams to Enhance Text Categoriation. – Information Prrocessing & Management, 2002, 38 p, DOI: 10.1016/S0306-4573(01)00045-0

50. Yang, Y., Pedersen, O. J. A Comparative Study on Feature Selection in Text Categorization. – ICML, Vol. 97, 1997, pp. 412-420, URL: www.surdeanu.info/mihai/-teaching/ista555-spring15/readings/yang97comparative.pdf

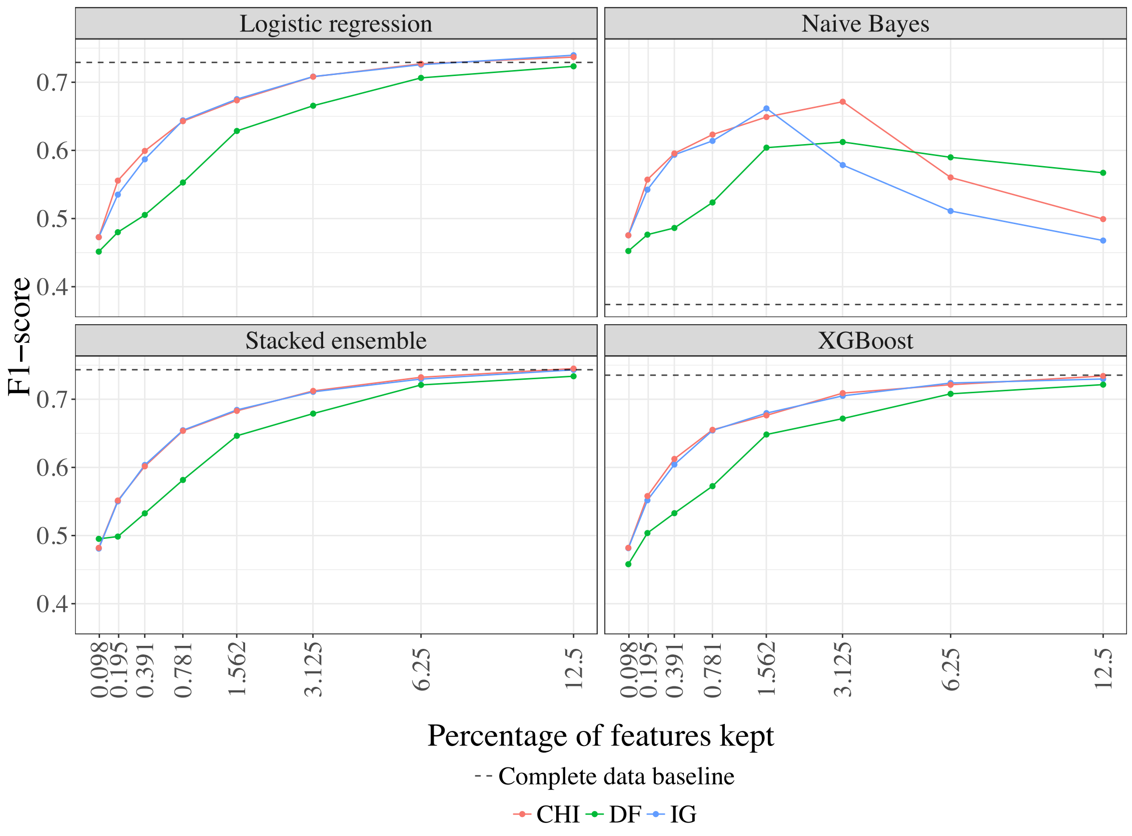
51. Zhang, D., Zhou, L. Discovering Golden Nuggets: Data Mining in Financial Application. – IEEE Transaction on Systems, Man, and Cybernetics – Part C: Applications and Reviews, 2004, Vol. 34, No. 4, pp. 513–522, DOI: 10.1109/-TSMCC.2004.829279

52. Zhang, Y., Jin, R., Zhou, Z. H. Understanding Bag-of-Words Model: A Statistical Framework. – International Journal of Machine Learning and Cybernetics, Issue 1-4, Vol. 1, 2010, pp. 43-52, DOI: 10.1007/s13042-010-0001-0

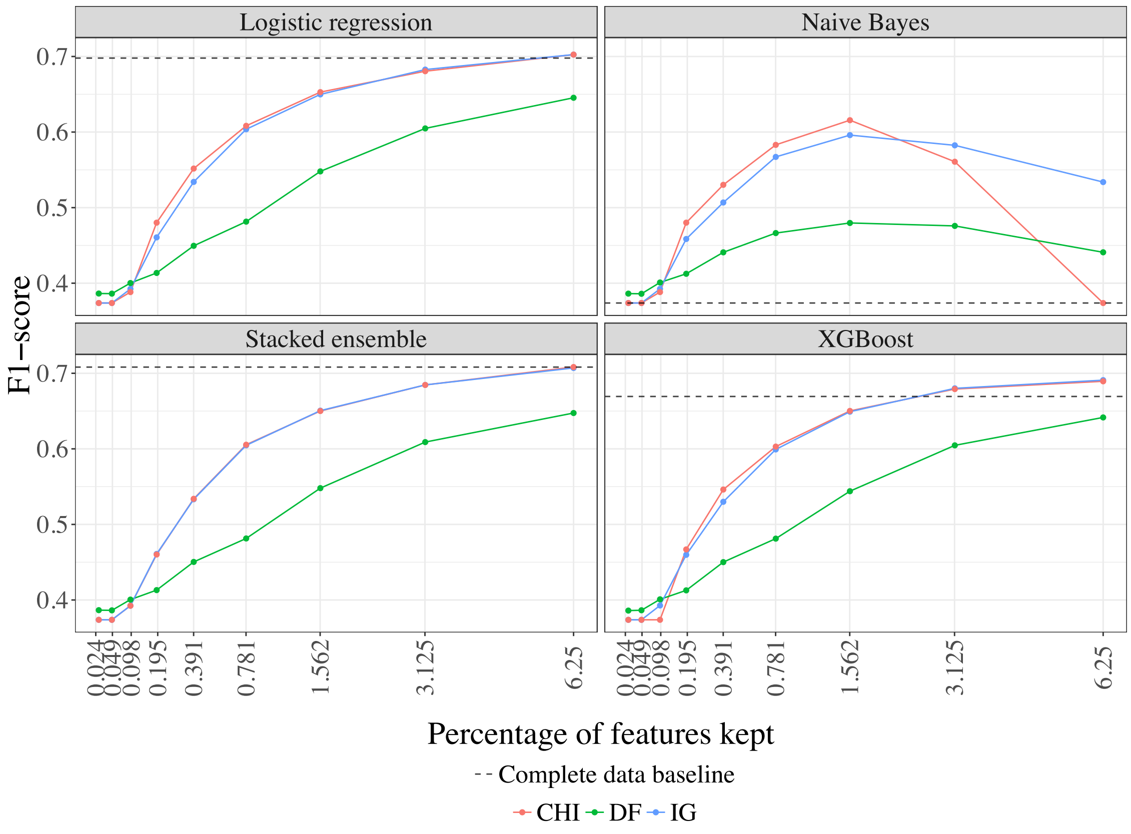
53. Zhu, J., Xie, Q., Zheng, K. An improved early detection method of type-2 diabetes mellitus using multiple classifier system. – Information Sciences, Vol. 292, 2015, pp. 1-14, DOI: 10.1016/j.ins.2014.08.056

Appendices

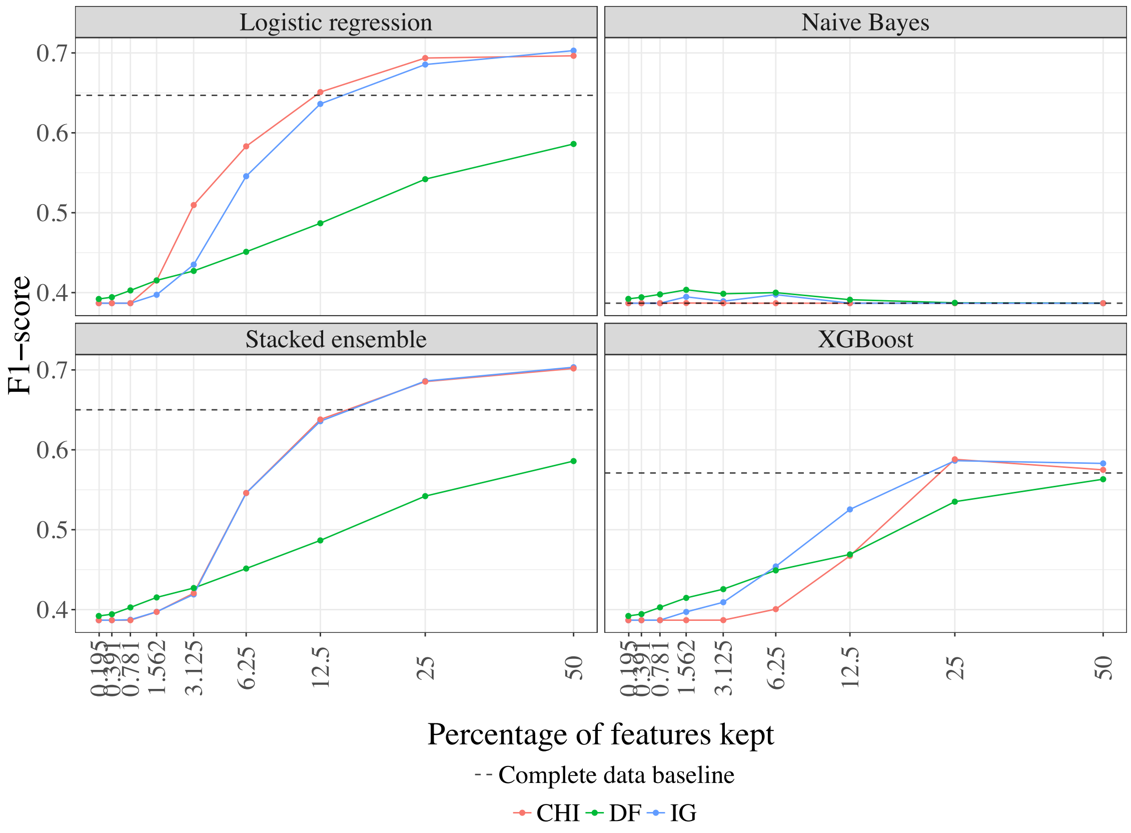
Appendix 1. Unigram classifier performances by feature selection method using term frequency weighting



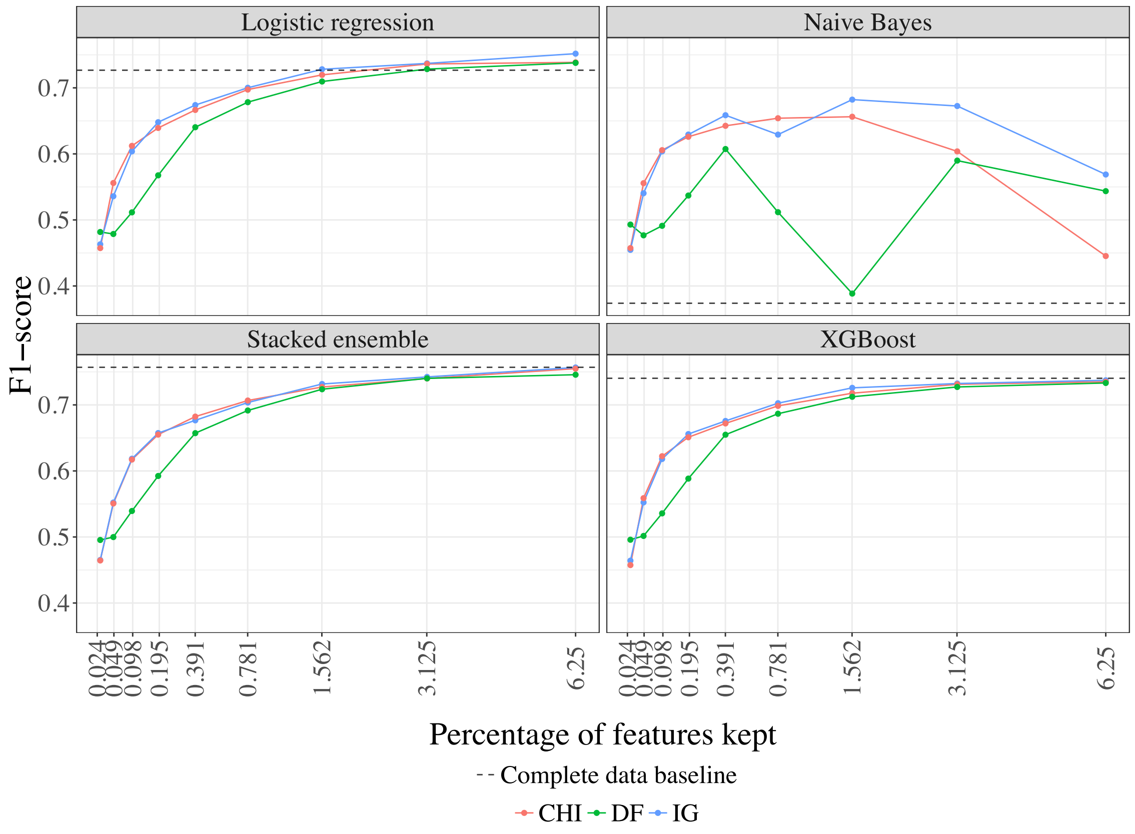
Appendix 2. Bigram classifier performances by feature selection method using term frequency weighting



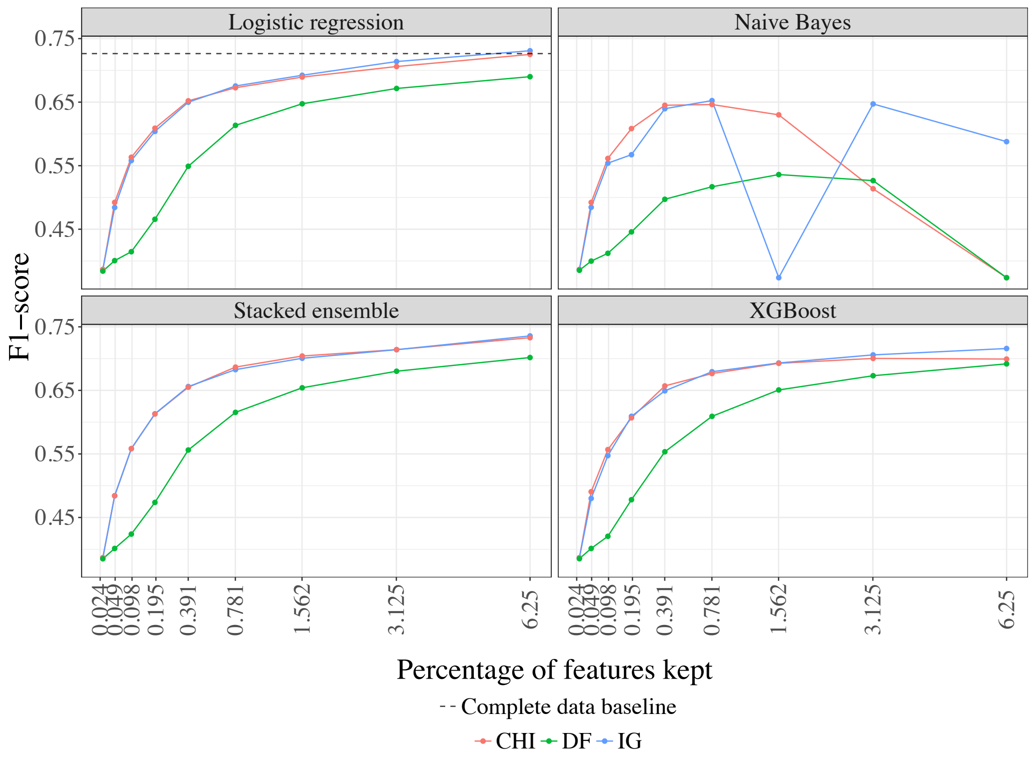
Appendix 3. Trigram classifier performances by feature selection method using term frequency weighting



Appendix 4. Uni-, bi- and trigram classifier performances by feature selection method using term frequency weighting



Appendix 5. Two-skip-bigram classifier performances by feature selection method using term frequency weighting


Logistic regression
Naive Bayes
Stacked ensemble
XGBoost
F1–score
Percentage of features kept
Complete data baseline
CHI
DF
IG