TEXT CLASSIFICATION

CLASSIFYING EVENTS TO UGENDA CALENDAR GENRES

Submitted in partial fulfillment of
the requirements for the award of the degree of

Bachelor of Science
in
Artificial Intelligence

Submitted by

Tanja Crijns s4204999

Supervisors:

Dr. L. G. Vuurpijl AI Department
Dr. F. Grootjen AI Department
Dhr. S. Trooster Ugenda

Radboud University
Department of Artificial Intelligence

ABSTRACT

Ugenda is a leading cultural event website that faces a challenge in the information management of their event calendar. The process of verifying and preparing information from venues is time-consuming and the team is looking for a way to automate this process. Certain event details are often missing, such as the event genres. These are important for sorting the calendar. This thesis proposes a solution for automatically labeling events that lack a genre. The focus is on three subjects; *event details*, *pre-processing techniques* and *classification methods*. We try to find a combination that works well enough for an operating website. The pre-processing methods included *natural language processing*, *HTML tag removal*, *date, time and location feature mapping*. The four classifiers used were *support vector machines*, *logistic regression*, *naïve bayes* and *random forest*. Results show that the logistic regression classifier has the best performance with a complete setup of proposed pre-precessing methods and event details. An F1-score of 0.8110 was achieved, which is not enough for an operating website.

CONTENTS

1 INTRODUCTION 1
1.1 The automation of Ugenda . . . 1
1.2 Reducing workload by automation . . . 1
1.3 Text classification . . . 2
1.4 The multi-label aspect of classification . . . 3
1.5 Research aim . . . 3

2 MULTILABEL TEXT CLASSIFICATION 5
2.1 Introduction to text classification . . . 5
2.2 Pre-processing . . . 5
2.2.1 Data transformation . . . 5
2.2.2 Feature selection . . . 7
2.3 Common text classification issues . . . 7
2.3.1 Overfitting . . . 7
2.3.2 Imbalanced data . . . 8
2.4 Multi-label classification . . . 8
2.5 Choosing a classifier . . . 9
2.5.1 Support vector machines . . . 10
2.5.2 Logistic regression . . . 10
2.5.3 Naïve Bayes . . . 10
2.5.4 Random forest . . . 11

3 DATASET 13
3.1 Physical attributes . . . 13
3.2 Selecting relevant event components . . . 14
3.3 Dataset challenges . . . 14
3.3.1 Duplicate events . . . 14
3.3.2 Imbalanced genres . . . 15

4 METHODOLOGY 17
4.1 Programming language and tools . . . 17
4.2 Implementation . . . 18
4.2.1 Data handling . . . 18
4.2.2 Pre-processing . . . 20
4.3 Classification . . . 21
4.3.1 Vector representation . . . 21
4.3.2 Classifiers . . . 21

5 RESULTS 23
5.1 Performance measures . . . 23
5.2 Pre-processing technique results . . . 24
5.2.1 Natural language processing . . . 24
5.2.2 Date and time feature mapping . . . 24

5.2.3 Location feature mapping 24
5.3 Classification results 25

6 CONCLUSION 27
6.1 Main findings 27
6.1.1 Pre-processing techniques 27
6.1.2 Classification 28
6.1.3 Main performance evaluation for an operating website 28
6.2 Future work 29
6.2.1 N-grams 29
6.2.2 Dataset imbalance 29
6.2.3 Toleration for similar category misclassification 30

BIBLIOGRAPHY 31

1

INTRODUCTION

1.1 THE AUTOMATION OF UGENDA

A few decades ago, one would watch the eight o'clock news or read the morning newspaper to get an update on world affairs and local events. Nowadays, the internet is often consulted. The internet has become one of the primary sources of information. People share and obtain a large amount of information online every day. The Ugenda website[1] makes use of this fact. Ugenda is a leading cultural event website. It is a platform for everyone that is involved with organising (cultural) activities in Nijmegen. The website hosts an event calendar and posts articles, discussions and reviews about these events.

Ugenda faces a challenge in information management of the event calendar. The content of the Ugenda website is maintained by a team of web developers. They receive event information from venues or groups that host cultural events in all shapes and sizes. This information is received in a document by mail or acquired from the organization's website. The process of verifying and preparing this information for publication is time-consuming. The Ugenda team is looking for a way to automate this process. In this thesis, a subproblem of this process will be covered; the automation of classifying events to Ugenda calendar genres.

1.2 REDUCING WORKLOAD BY AUTOMATION

Automation of an information handling process requires a high percentage of correct output. If the Ugenda team has to analyze and correct the majority of the produced output, it is questionable whether the automation is useful. In order to get a low number of errors, identification of problems that could cause faulty output is needed. For instance, when processing received event information, missing certain details is a problem. The Ugenda team can often fill in these missing details by using their common sense. This logic needs to be implemented in the automated program so that it can handle this as well. The events in the Ugenda event calendar contain the following details: *date*, *time*, *title*, *location*, *description* and *genre*. Most of the time, these details are provided by the venues or groups. In the other cases, the team scrapes the required information from a website. In particular, the genre is often missing. The genre is a category such as *kids*, *music*

1 https://www.ugenda.nl/

or *dance*. If a scraped event ends up in the database without a genre, it will not show in the calendar on the website. In this thesis we explore the feasibility of automatically assigning this genre.

1.3 TEXT CLASSIFICATION

Text classification is a widely studied subject in the information science sector. It has been relevant ever since the origin of digital text documents. Text classification generally involves a multi-step process. Ikonomakis et al.[1] show the text classification process in a simple graph (see Figure 1).

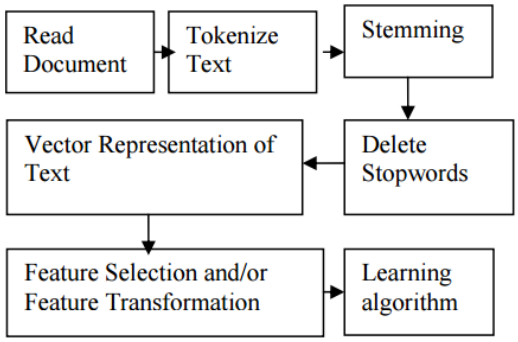


Figure 1: The text classification process

The text classification process can roughly be divided in two sections: *pre-processing* and *classification*.

The *pre-processing* section roughly contains step one through six (see Figure 1). The main goal of the whole classification process is to assign a label to an event by evaluating its textual details, this label corresponds with an Ugenda calendar genre. However, it is hard to compare a large bulk of textual details from one event to that of another event without breaking both bulks down into smaller pieces. A text can be separated into a set of words, which we will use as features for classification. The performance of classification generally improves if the features are transformed and filtered, which reduces the size and sparsity of the feature set.

The final step in Figure 1 is classification. In this step, a classifier is used to predict a label for unlabeled events. A model is built based on labeled training data and this model will predict the label of unlabeled, previously unseen test data.

The difference in the size of these two sections might appear unusual as the actual classification seems to be an important part of the whole process. However, the performance of a classifier is mainly determined by the quality of the used feature set. Uysal and Gunal[2] show that

choosing suitable pre-processing techniques can provide a significant improvement of the classification performance. They compared many different combinations of widely used pre-processing techniques on two domains. They concluded that which pre-processing techniques are suitable depends on the domain, language and task. A brute force approach such as this one will definitely yield results. However, focusing on why certain techniques do or do not work will provide a deeper understanding of the problem.

1.4 THE MULTI-LABEL ASPECT OF CLASSIFICATION

Events on the Ugenda website are distributed over the ten genres presented in Table 1.

Table 1: Ugenda calendar genres

Genre	English translation
Beeldend	Visual art
Dans	Dance
Evenement	Event
Film	Film
Kids	Kids
Muziek	Music
Party	Party
Theater	Theatre
Woord	Word
Varia	Miscellaneous

An event can ecompass multiple genres, this makes it a multi-label problem. Most traditional algorithms are based on single label, binary problems. Common approaches to multi-label problems will be discussed in Chapter 2.

1.5 RESEARCH AIM

The research was based on a main research question which consists of three sub-questions. The main research question reads:

In classifying events to Ugenda calendar categories, which event details and which combination of data preparation and classification techniques will perform well enough to be usable on an operating website like Ugenda?

The three sub-questions read:

- *Are title and event description enough for a satisfactory performance or are other event details needed?*

- *What methods of data preparation are most useful?*
- *Which classification techniques are most useful and how do they perform compared to each other?*

The answers to these questions will be formalized in Chapter 6.

2

MULTILABEL TEXT CLASSIFICATION

In this chapter, an overview of text classification will be given. We will start with a general introduction. After that, the pre-processing procedure will be explained and common classification issues will be reviewed. Lastly, the choice of classifiers and their function will be discussed.

2.1 INTRODUCTION TO TEXT CLASSIFICATION

The goal in automatic text classification is to assign a document to a category by evaluating its text components. It has been applied successfully multiple times and is integrated in our everyday lives. For instance, newspaper articles and academic papers are often organized by subject or field. Text classification provides a solution in automatically organising countless articles and papers. Another well known application is spam filtering. It is becoming more and more common to receive unsolicited emails daily. By using text classification to label these emails as spam, they can be filtered automatically. Sculley et al.[3] achieve the following results; ≈0.88 accuracy, ≈0.91 precision, ≈0.85 recall, in spam filtering by using two variants of Support Vector Machines. Another interesting application of text classification is automated threat detection in social media. Kandias et al.[4] use text classification to detect negative attitudes towards law enforcement. They classify comments to two categories, one with negative attitude comments and one with neutral attitude comments.
As explained in Chapter 1, the text classification process can be divided in two sections: *pre-processing* and *classification*. Pre-processing will be explained next.

2.2 PRE-PROCESSING

Before one can build a classification model, it is necessary to transform the data into useful features. The first six steps of Figure 1 will be used as reference for the pre-processing process. The first five steps are considered *data transformation*, which will be discussed next.

2.2.1 *Data transformation*

Tokenization is the first important step in the transformation of the dataset. Before we can use a text for classification, we need to able to identify the separate features of a text. This can be done in different

ways. For example, we can split the text into a set of words or we can split it into a set of word sequences. These words or word sequences will be will be referred to as tokens hereafter.
Splitting the text into a set of words is done by searching for specific token separators. Two examples of separators that are easy to determine are white spaces and newline indicators like \n. However, some separators are ambiguous, such as a dot. It can be part of a token when it is in a number such as 35.0, but it can also indicate the end of a sentence.

Splitting the text into a set of words sequences is called n-gram processing. An n-gram is a continuous sequence of n entities of a text. Entities could be text elements like words, syllables or characters. The benefit of using n-grams is the gain of more context in a single token and that information from word order could be disclosed. Some words are more meaningful when combined with a second or even a third word. For example, the word 'advisory' is more meaningful when it is combined with the word 'board' . Abou-Assaleh et al.[5] show that using n-grams can be beneficial for text classification performance when using up to three entities per token.

Stemming is an example of *natural language processing*. There are other methods of language processing that can be used, but stemming is the most common. Stemming means transforming words with approximately the same semantics to their standard form. For example, reducing all words in plural form to singular form. Another option is to transform all words in past tense to present tense. Examples of other methods of language processing are text simplification or adding synonyms of the words to the token set.

Deleting stop words can reduce the size of the feature set without performance loss. Stop words like *de(the)* and *een(a)* are likely to have no predictive value.

Making a vector representation of a text means transforming the feature set of tokens into a dictionary where numbers represent words. A classifier can only work with numbers and not with the textual representation of a word. There are several ways to vectorize, vectorization by *count* and *tf-idf* will be discussed[6].
A *count vectorizer* determines the value of a token based on its frequency. This is called *term frequency*. It keeps track of how many times all seperate tokens occur in a text. The more often it occurs, the higher the value of the token is.
Tf-idf takes it one step further. It considers the term frequency, but also takes the token specificity into account. This combination is called *term frequency - inverse document frequency*. For example, the token 'the'

often occurs with high frequency in all texts. A tf-idf vectorizer will assign a low value to this token. A word that occurs often in few texts but less in other texts will receive a higher value.

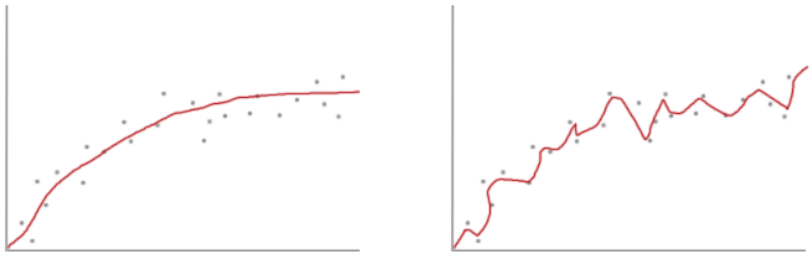
2.2.2 *Feature selection*

Feature selection is a way of finding the best possible feature set. The number of noisy or unspecific features is reduced. Reducing the feature set to a smaller set with specific features can prevent overfitting. It is also important when the feature set is too large to handle. This means that processing and classifying the entire feature set takes such a large amount of time that it is hard to work with. If this is the case, performing dimensionality reduction can reduce the training time of the algorithm without performance loss[7]. An example would be to filter the words that occur often in all texts of the dataset. These words do not contribute much to the performance of classification but still consume resources.

2.3 COMMON TEXT CLASSIFICATION ISSUES

2.3.1 *Overfitting*

Overfitting occurs when the model learns the training data in a too literal manner. It should actually learn the general patterns that can be found in the data. It prevents the model from correctly predicting unseen data. For a visual representation, see Figure 2.



(a) Right fitting by the model (b) Overfitting by the model

Figure 2: Visual representation of overfitting

There is no general solution to overfitting because there are many reasons as to why this can happen. A solution that is often applicable is to collect more data. In cases where this is not possible, the origin of the overfitting problem and its corresponding solution should be determined. There is however a way to detect overfitting by using cross validation[8]. Cross validation is a technique of partitioning the training and test data. This is different from just having fixed percent-

ages of training and test data. An example is to use the k-fold method. When using k-fold cross validation, the dataset is partitioned into k equally large subsets. One of these subsets is declared test set and the remaining sets form the training data. This process repeats itself k times with other test- and training sets. Every set in the training data is then used to produce a prediction and these results are combined. The results represent how well the model generalizes to new data. If the cross validation score is very high, it is likely that the model is overfitting.
For small data-sets, cross-validation can provide some improvement in overfitting. If there is little data, a classifier is likely to overfit due to a high variance, which will be explained in Section 2.5. Cross-validation uses all data to train and test. This means that the model is trained on more data than when there is a certain percentage of train and test data. This reduces variance and can lower overfitting.

2.3.2 *Imbalanced data*

Real world datasets often lack an even distribution. Some classes may have more data points than others. This should be taken into account as the classifier might develop a bias towards the larger classes. With a biased classifier, it could seem that the classifier achieves a good performance. For instance, there are two classes x and y. Class x has 80 data points and class y 20. If the classifier were to develop a bias for class x and classifies everything as that class, an accuracy of 0.8 would be achieved. This seems to be a good performance even though all instances of the class y are wrongly predicted. There are several methods to tackle this problem. One option is to resample the dataset in order to get equall sized classes. When there is a small dataset, *over-sampling* is used. This means that copies of instances of small classes are added tot the dataset. However, having duplicates in the data could cause overfitting.
When there is a large dataset, *under-sampling* is used. This means that instances of large classes are deleted. When the dataset does not allow for under-sampling or over-sampling, there are other methods such as reinforcing different misclassification costs. With this method, misclassifying a large class will have a lower cost than misclassifying a smaller class. This way, the classifier is encouraged to grant a label of the smaller class more often.

2.4 MULTI-LABEL CLASSIFICATION

As mentioned in Chapter 1, most traditional text classification algorithms are based on binary problems. In a binary setup, an instance from the dataset can receive only one label. A common approach is to convert the multi-label problem into multiple separate binary prob-

lems. This has been applied successfully. For instance, Dhillon et al.[9] apply this by using a *one-versus-rest* method and achieve satisfactory results. In this method, one binary classifier is trained per label, as opposed to training a single classifier for all labels. This solution is independent of the algorithm used for classification.

2.5 CHOOSING A CLASSIFIER

A classifier is an algorithm that can predict the labels of unseen data. There are many different kinds of classifiers that are suitable for different problems. Choosing the right one is crucial for the performance of the program. The choice mostly depends on the shape and size of the dataset. A classifier's error in classification is defined as the sum of the *bias* and the *variance*.

The *bias* of a classifier is low if the model represents the true data distribution well. It does not depend on training set size. A high bias can cause the classifier to underfit, which means that the model misses important relations between features.

The *variance* of a classifier is low if there are few fluctuations in the training set. It depends on the training set size. A high variance can cause the classifier to overfit, which means that the classifier models random noise in the dataset instead of the actual relations.

With a small dataset, the variance is often high. In this case, a classifier that can deal with high variance like Naïve Bayes is a good choice. This classifier disregards the high variance by processing all features independent from each other. When the dataset is bigger, the shape of the data and the scope of the problem becomes important. The scope of our problem is text classification. Sebastiani[10] evaluated different classifiers for text classification and found the following:

1. Boosting-based classifier committees, support vector machines, example-based methods, and regression methods deliver top-notch performance.
2. Neural networks and on-line linear classifiers work very well, although slightly worse than the previously mentioned methods.
3. Batch linear classifiers and probabilistic Naïve Bayes classifiers perform the worst of the learning-based classifiers.
4. The data is insufficient to say anything about decision trees.

The dataset on which the program will be trained is relatively small. The shape and size of the dataset will be detailed further in Chapter 3. Based on the findings of Sebastiani[10] that were presented earlier, four classifiers have been selected for comparison:

1. Support vector machines
2. Logistic regression
3. Naïve Bayes
4. Random forest

Support vector machines and *Logistic regression* were chosen as they performed the best in Sebastiani's research. Secondly, *Naïve Bayes* was chosen as baseline. Even though it performed worst in Sebastiani's research, the dataset is small and *Naïve Bayes* works well with high variance. As the fourth classifier, the *Random forest* classifier was chosen in order to see how a decision tree based classifier performs on the dataset. On what principles these classifiers are based and how they work will be explained next.

2.5.1 *Support vector machines*

A support vector machine[11] is a binary classifier from the outset. Binary means that the classifier divides objects into two classes. In classification, this means that it determines whether the object belongs to the class or not. In order to do that, values are mapped in a hyperplane. A linear function is used to determine a boundary that divides the objects into two classes. The boundary is based on the distance to the closest objects in either class. This distance needs to be maximized. The closest objects are called the *support vectors*.

2.5.2 *Logistic regression*

A logistic regression model is a binary classifier. It is similar to linear regression, where the task is to ascertain a value for each object in the dataset. In contrast to the outcome of linear regression, the outcome of logistic regression is not continuous but binary. A logistic function is used to determine the probability of a value belonging to the class or not.

2.5.3 *Naïve Bayes*

A naïve Bayes classifier is a binary classifier. It is based on Bayes' theorem:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

This theorem shows a way to find the probability of A given evidence B. In classification, this means that we have an object A and category evidence B. A certain evaluation method such as a threshold is chosen to determine whether the objects belongs to the class or not. For

example, if the probability is higher than 0.6, it belongs to the category. This classifier considers all features to be independent.

2.5.4 *Random forest*

A random forest classifier averages multiple decision trees based on random samples from the database. A decision tree breaks the dataset down into smaller subsets while simultaneously building a tree with decision nodes and leaf nodes. A decision node has two or more branches with choices or leafs. A leaf node represents a category. Because the trees in a random forest are based on random data, they can be noisy and lack meaning. The random forest averages these trees to create a model with low variance. The irrelevant trees cancel each other out and the remaining meaningful trees yield the result.

3 DATASET

In this chapter, the Ugenda dataset will be depicted. Firstly, the *physical attributes* of the dataset will be discussed. After that, specific *components* of the events will be evaluated by utility. Lastly, challenges specific to this dataset and solutions to them will be discussed.

3.1 PHYSICAL ATTRIBUTES

Ugenda has provided an SQL database which contains 12.675 events. The events in the database are distributed over 12 genres although there are only 10 actual genres on the Ugenda website. The database also contains a lot of duplicate events. After removing these duplicates and the events in non-existing genres, the dataset size was reduced to 6.803 events. In Section 3.3.1, the removal of duplicates will be discussed. Each event in the database contains at least the following information: *id number*, *location*, *start date*, *start time*, *title*, *event description*. Events may also contain information in custom fields like *end date*, *end time* or *entrance fee*. However, these custom fields are not consistent across all events and are often left blank. Table 2 shows an example of the raw content of an event from the database. Only few useful information slots are left blank in this specific event, which is quite rare in the dataset.

Table 2: Example event

Event component	Content
ID	20146
Location	Valkhofpark
Start date	2014-06-04
Start time	12:00:00
End time	22:00:00
Title	Valkhof Klassiek
Event description	Op 1 juni 2014 vindt de eerste editie van Valkhof Klassiek plaats: een dag lang klassieke muziek in het Valkhofkwartier te Nijmegen. Overdag musiceren klassieke ensembles op verrassende binnen- en buitenlocaties. In de namiddag en vroege avond treden grote orkesten op in het Valkhofpark. 's Avonds wordt er klassiek gemusiceerd bij restaurants in het Valkhofkwartier.
Website	http://valkhofklassiek.nl/

3.2 SELECTING RELEVANT EVENT COMPONENTS

A single event contains a lot of components holding information. However, not all of these components are useful for classification. Which components are used for classification will be discussed next. The first two selected components are the title and the event description, as it is evident that these may contain useful information. The next two components are date and time. Besides information on the specific timing of an event in the year, the date also holds information about on which day in the week the events takes place. These two components can be useful because it can help classify prominent genres. For instance, an event of the genre *kids* generally takes place at very different times than an event of the genre *music* or *dance*. The last component is the *location*. It could hold useful information; some venues only hold events of a single genre.

3.3 DATASET CHALLENGES

The provided dataset was a raw database which was not altered for this thesis by Ugenda. Two challenges arose when inspecting the database; the many *duplicate events* and the *imbalanced categories*.

3.3.1 *Duplicate events*

More than half the events in the initial dataset are duplicates of another event. The high number of duplicates is caused by the recurring nature of some events, such as movies. An event is considered a duplicate if title and event description are equivalent but date or time is different. There are arguments in support of removing these duplicate events, and against. Arguments of both sides will be considered. The main argument for duplicate removal is the high risk of overfitting. When the training set contains a lot of duplicate events, it is likely that this is considered an important connection during training. The model is fit to represent the duplicate events well and this causes a higher classification error for less frequent events. However, the number of duplicates in the set is a good representation of the real world situation. Movies simply occur more often than yearly or non-recurring events. In order to get the purest performance, the duplicates were removed. It is however uncertain whether this works better in practice. This requires a different research direction so it will not be discussed further.

3.3.2 *Imbalanced genres*

The distribution of events across the genres can be seen in Table 3. Overall, there is a clear imbalance. The music genre has the most events and the dance genre has the least. The risk of an imbalanced dataset was explained in Section 2.3.2, the solutions that were proposed will be considered for this dataset next. Because this dataset is small, that means that it has less than 100.000 events[1], it is not wise to under-sample. Under-sampling with little data causes too much information loss. Because of the overfitting risk that comes along with over-sampling, this is not wise either. The remaining option mentioned is reinforcing a higher cost of the misclassification of small genres.

Table 3: Number of events per genre

Genre label	English translation	Frequency
Beeldend	Visual art	418
Dans	Dance	121
Evenement	Event	943
Film	Film	227
Kids	Kids	369
Muziek	Music	2914
Party	Party	592
Theater	Theatre	696
Woord	Word	696
Varia	Miscellaneous	296
Total		7272

1 http://scikit-learn.org/stable/tutorial/machine_learning_map/

4 METHODOLOGY

In this chapter, an overview of the used *programming language* and *tools* will be given. After that, it will be explained how the *dataset* was *handled* and how the previously mentioned *pre-processing* and *classification techniques* were implemented. The process flow is shown in Figure 3. Seperate components of this flow diagram will be discussed.

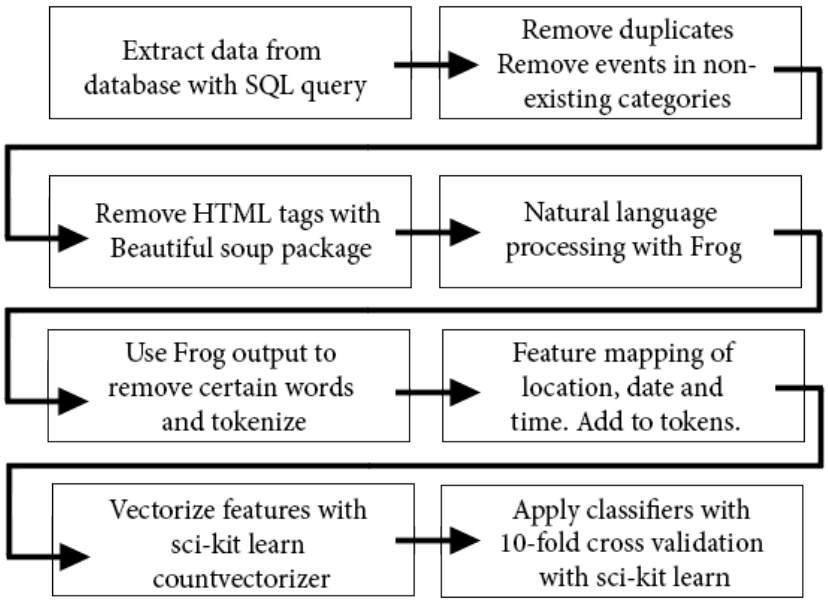


Figure 3: Process flow

4.1 PROGRAMMING LANGUAGE AND TOOLS

Most of the applied machine learning methods used implementations from the Python library scikit-learn[12] and the Dutch natural language processing system Frog[13]. Scikit-learn was used with version 2.7 of Python and Frog with version 3.3.4 of Python. A git repository with all code used in this thesis can be found on Gitlab[1].

1 https://gitlab.socsci.ru.nl/tanja987.crijns/mltc-Ugenda

Scikit-learn is a free machine learning software library which is designed to cooperate with other frequently used Python libraries like Numpy[14] an SciPy[15]. It focuses on modelling data and provides multiple supervised and unsupervised learning algorithms. It also has many methods that contribute to the modelling of data like cross-validation and feature selection.

Frog is a collection of Dutch natural language processing modules. This has everything to do with pre-processing. The current version is able to perform tokenization, lemmatization and part of speech(PoS) tagging among other functions.

4.2 IMPLEMENTATION

4.2.1 *Data handling*

The data was extracted from the SQL database and represented in a dictionary data structure. The ID of an event indicates the position in the dictionary. An event is represented as a list containing multiple information components. Duplicate events were removed by comparing all titles and event descriptions. If the title and description of an event matc with those of an event that is already in the dictionary, it is discarded.

The event description and the title were first focus. Appropriate columns were selected from the dictionary and combined in a string, which was be processed by *Frog*. This will be explained in Section 4.2.2.

Event starting time, starting date and location were added later. The extraction of these components was less straightforward than the extraction of the event description and title. Generalized times, dates and locations provide more information than specific ones. For example, two events that are both music performances should be classified to the *music* category. The fact that one event is held on Friday at 20:00 and that the other is on Saturday at 21:00 shows nothing in common even though both are music performances that happen in the evening, on the weekend. Generalizing the starting date and time to *weekend* and *evening* provides a common factor.

The *starting time* of an event is mapped to a time feature which has three possible values: the *morning* value if the event started between 00:00 and 12:00, the *afternoon* value if the event started between 12:00 and 18:00, the *evening* value if the event started between 18:00 and 24:00 and the *unknown starting time* value if the event lacks a proper starting time.

The starting date of an event is mapped to a date feature which has three possible values: the *weekend* value if the event starts on Friday, Saturday or Sunday, the *weekday* value if the event is on any other day of the week and the *unknown starting date* value if the events lacks a proper starting date.

The location of an event is mapped to a location feature which has ten possible values. The dataset contains 748 different locations. Not all locations have a relevant name. For example, if an event is held in a bar, it does not matter what that bar is called. This also applies to libraries, art galleries, museums and theatres. How this was generalized can be seen in Table 4. The terms in the second column were hand-picked after studying the locations in the database. Besides these possible values, it is also possible to map the location to the *other location* value when the location id is not correct. This is included because there were some cases in which the location id did not refer to any of the locations in the database. If the location id is correct but the location is not mapped to each of the 8 values in Table 4, it remains the same.

Table 4: Featurem apping of location

Location feature value	Location name contains
School	school, college, universiteit, campus, gymnasium
Bar	bar, café
Library	bibliotheek, library
Outside	park, plein, kade, tuin, buiten, berendonck
Cinema	cinema, bioscoop, filmhuis
Theatre	theather, schouwburg, toneel
Factory	fabriek, honig, vasim

Title, event description, starting time feature, starting date feature, original location and location feature were concatenated to form the plain text, which is the input for vectorization. The latter four were added twice. The reason for this is that in the title and event description text combination, the general word count is greater than one.

4.2.2 *Pre-processing*

The first adjustment to data was the removal of HTML tags. Some event descriptions still contained HTML tags, which was also the case for the event in Table 2. A Python package called *Beautiful Soup*[2] was used to remove HTML tags.
Following the standard steps of text classification; *Tokenization*,*stemming* and *stop word removal* were implemented with the help of *Frog*. The functions *tokenization*, *lemmatization* and *part of speech tagging* were used. Although lemmatization resembles stemming, it is not equivalent. In stemming, one normalizes a word without knowledge of the context of that word. Lemmatization implies a broader scope of normalization. In lemmatization, one tries to find the normal dictionary form of a word based on grammar and context. Lemmatization was chosen because this is provided by Frog. If we take the event presented in Table 2 as an example, the Frog output would look like Table 5. The first column indicates the token number of the feature in its sentence, the second column represents the lemma of that feature and the third shows the part of speech tag.

Table 5: First few lines of Frog output of the event with id 20146

#	Token	Lemma	Part of speech tag
1	Valkhof	Valkhof	SPEC(deeleigen)
2	Klassiek	Klassiek	SPEC(deeleigen)
3	Op	op	N(soort,ev,basis,zijd,stan)
4	1	1	TW(hoofd,vrij)
5	juni	juni	SPEC(deeleigen)
6	2014	2014	TW(hoofd,vrij)
7	vindt	vinden	WW(pv,tgw,met-t)
8	de	de	LID(bep,stan,rest)
9	eerste	een	TW(rang,prenom,stan)
10	editie	editie	N(soort,ev,basis,zijd,stan)

Frog has 12 possible part of speech tags, which can be found in Table 6. Frog bases these tags on the '*Corpus Gesproken Nederlands*' tagset[16]. Stop words can be removed based on these tags. For instance, every token that has the 'LID' tag, which are words like 'the' and 'a', can be removed. For this dataset, it was useful to remove tokens with the 'LID', 'VZ', 'VG' and 'BW' tags. Stop words still occurred in the other unremoved tags, so a list of common dutch stop words[3] was filtered in the plain text as well.

2 https://www.crummy.com/software/BeautifulSoup/
3 http://web.tue.nl/bib/w/vubissmart/foto/foto-stopwoorden.html

Table 6: Frog part of speech tags

Tag	Part of speech
ADJ	Adjective
BW	Adverb
LET	Punctuation
LID	Determiner
N	Noun
SPEC	Names and unknown
TSW	Interjection
TW	Numerator
VG	Conjunction
VNW	Pronoun
VZ	Preposition
WW	Verb

4.3 CLASSIFICATION

4.3.1 *Vector representation*

A k-fold cross validation method of *scikit-learn* with ten fold was used to obtain training- and test sets. In order to compare the results, it was possible to provide a random state so that the input remained the same. The count-vectorization method was used to transform the sets into feature vectors, this method is based on *term frequency*.

4.3.2 *Classifiers*

Scikit-learn implementations of *support vector machines*, *logistic regression*, *naïve Bayes* and *random forest* were used for classification. The classifiers were applied in a one-versus-rest manner, which makes it possible that no label is assigned to an event. The Ugenda event calendar however requires a category. If no label was originally assigned, the most likely label was assigned instead.

5 RESULTS

In this chapter, we will first discuss how the results are presented by explaining the performance measures. After that, the final results in Table 8 will be explained across the sections Section 5.2 and Section 5.3

5.1 PERFORMANCE MEASURES

In classification, there are many perfomance measures. Four common measures will be discussed.

Accuracy measures how often the classifier makes the correct prediction. This is however not very insightful because of the imbalanced dataset. We can look at the following example: if we have a dataset with 100 objects of which 90 belong to category 1 and the remaining 10 belong to category 2. If we assign everything to category 1, an accuracy of 0.9 would be achieved even though category 2 is predicted completely incorrect.

Recall, *precision* and the *f1-score* are better suited for this dataset. The recall score is calculated as the number of times the classifier assigns a label of a specific category out of all the existing labels of that category. The precision score is calculated as the number of times a label of a specific category is assigned correctly. The F1-score is a trade-off between these two scores.

Standard recall, precision and F1-score are defined for single-label classification problems. Because our problem is multi-label, we have to combine the values of every individual classifier. We will use an altered version of the recall, precision and F1-score. The averaging of recall and precision across labels will not be standard, but adjusted with weights based on support. This could cause an F1-score that is not in between recall and precision. We have an imbalanced dataset and these measures take the imbalance into account.

5.2 PRE-PROCESSING TECHNIQUE RESULTS

5.2.1 *Natural language processing*

With the word *processed* in Table 8, rows *B*, *D*, *F* and *H*; HTML tag removal, stop word removal and natural language processing with Frog is meant. The results can also be found in these rows. HTML tag and stop word removal improved performance across classifiers and feature set compositions. The definitive configuration of the natural language processing was also based on the highest performance across classifiers and feature set compositions. An example of a different configuration was that lemma's were used instead of tokens or that different part of speech tagged words were removed.

5.2.2 *Date and time feature mapping*

The final representation of starting date and starting time of an event was based on the highest performance across classifiers and feature set compositions. An example of a different representation was transforming the starting date into separate days or the starting time in to four hour time intervals. The results can be found in Table 8, rows *C*, *D*, *G* and *H*.

5.2.3 *Location feature mapping*

The definitive representation of location of an event was based on the highest performance across classifiers and feature set compositions. An example of a different representation was more possible values of the location feature, such as *art* when there was *art*, *gallery* or *exhibition* in the location name. The results can be found in Table 8, rows *E*, *F*, *G* and *H*

5.3 CLASSIFICATION RESULTS

Table 8 shows the performance of the four classifiers across the different feature set compositions. The feature set composition that yielded the highest F1-score is underlined for each classifier. The details of the highest F1-score can be found in Table 7

Table 7: Average performance per label across 10 folds and support in total dataset, logistic regression setup H in Table 8

Label	Recall	Precision	F1-score	Support
Beeldend	0.802	0.691	0.741	418
Dans	0.811	0.577	0.662	121
Evenement	0.688	0.64	0.661	943
Film	0.851	0.747	0.787	227
Kids	0.752	0.653	0.696	369
Muziek	0.858	0.931	0.895	2914
Party	0.833	0.791	0.808	592
Theater	0.774	0.822	0.796	696
Woord	0.78	0.763	0.776	696
Varia	0.579	0.435	0.492	296
Total	0.794	0.801	0.8110	7272

Table 8: Results across classifiers and feature set composition

Classifier	Recall	Precision	F1-score
Support vector machine			
A. Raw title and description	0.6908	0.7693	0.7090
B. Processed title and description	0.6836	0.7663	0.7226
C. Date, time, raw title and description	0.6950	0.7672	0.7055
D. Date, time, proc. title and description	0.6905	0.7712	0.7280
E. Location, raw title and des.	0.7266	0.7700	0.7363
F. Location, proc. title and des.	0.7310	0.7773	0.7508
G. Location, date, time, raw title and des.	0.7300	0.7842	0.7529
H. Location, date, time, proc. title and des.	0.7321	0.7832	0.7486
Logistic regression			
A. Raw title and description	0.7586	0.7679	0.7603
B. Processed title and description	0.7579	0.7684	0.7583
C. Date, time, raw title and description	0.7657	0.7715	0.7634
D. Date, time, proc. title and description	0.7651	0.7735	0.7598
E. Location, raw title and des.	0.7862	0.7962	0.8012
F. Location, proc. title and des.	0.7870	0.7984	0.8018
G. Location, date, time, raw title and des.	0.7938	0.8005	0.8012
H. Location, date, time, proc. title and des.	0.7942	0.8012	0.8110
Naïve Bayes			
A. Raw title and description	0.7474	0.7634	0.7041
B. Processed title and description	0.7503	0.7652	0.7083
C. Date, time, raw title and description	0.7412	0.7732	0.6974
D. Date, time, proc. title and description	0.7448	0.7739	0.7040
E. Location, raw title and des.	0.7646	0.7760	0.7225
F. Location, proc. title and des.	0.7669	0.7753	0.7242
G. Location, date, time, raw title and des.	0.7601	0.7830	0.7235
H. Location, date, time, proc. title and des.	0.7617	0.7828	0.7240
Random forest			
A. Raw title and description	0.7045	0.7493	0.7157
B. Processed title and description	0.7102	0.7572	0.7071
C. Date, time, raw title and description	0.7099	0.7578	0.6972
D. Date, time, proc. title and description	0.7163	0.7624	0.7220
E. Location, raw title and des.	0.7265	0.7710	0.7366
F. Location, proc. title and des.	0.7362	0.7807	0.7459
G. Location, date, time, raw title and des.	0.7371	0.7834	0.7424
H. Location, date, time, proc. title and des.	0.7395	0.7837	0.7303

6 CONCLUSION

In this chapter, answers to the following research questions, which were proposed in Section 1.5, will be formed based on the results showed in Chapter 5:

In classifying events to Ugenda calendar categories, which event details and which combination of data preparation and classification techniques will perform well enough to be usable on an operating website like Ugenda?

- *Are title and event description enough for a satisfactory performance or are other event details needed?*
- *What methods of data preparation are most useful?*
- *Which classification techniques are most useful and how do they perform compared to each other?*

6.1 MAIN FINDINGS

We will first form a partial conclusion to every separate section of the research questions, we will follow the sections formed in the results chapter(Chapter 5). The answer to the research questions can be found in Section 6.1.3.

6.1.1 *Pre-processing techniques*

6.1.1.1 *Natural language processing*

Overall, the natural language processing by Frog, HTML tag removal and stop word removal on title and event description improved the performance. The highest F1-score was achieved with this setup, which can be seen in Table 8. However, there are some cases in which the unprocessed title and event description yielded better results. For example, setup G and H for support vector machines or A and B for logistic regression. A possible explanation for this is that processing greatly improves certain features, but diminishes other features. If the feature set is altered so that the diminished features are lifted, than the processed feature set yields better results. If this is not the case, the unprocessed feature set yields better results. The unprocessed data has a more stable performance.

Furthermore, not all words that were tagged with a seemingly useless part of speech tag were removed in the best performance setup, which

was mentioned in Section 4.2.2. Also, lemmas were not used even though they provide more generalization than the original token. A possible explanation for this is that the dataset is too small to benefit a lot from generalization. Apparently, there is information in the conjugation of words or the use of for example, conjunctions. This could be caused by a writing style that is specific to people involved with events of a certain genre.

Even though the highest F1-score is achieved with natural language processing, the inconsistency in the results shows that natural language processing is not a reliable method for this specific dataset. An explanation for this could be that the dataset is too small. A bigger dataset requires more generalization and this is exactly what natural language processing provides.

6.1.1.2 *Date and time feature mapping*

Adding the date and time features mostly yields inconsistent results that either improve or decrease the performance. However, in setup F and H for logistic regression can be seen that the features have a significant impact on the the highest F1-score.

6.1.1.3 *Location feature mapping*

Adding the location feature generally yields significant performance improvement. In setup D and H for logistic regression can be seen that the features have a large impact on the highest F1-score.

6.1.2 *Classification*

The logistic regression classifier produces the best results, significantly. Support vector machines and the random forest produce classifier similar results and Naïve Bayes performs worst. This is in conformity with Sebastiani's[10] findings mentioned in Section 2.5.

6.1.3 *Main performance evaluation for an operating website*

Table 7 shows the average performance per label in cross validation for setup H with logistic regression. The genre with the largest support has the best performance. Other labels with a much smaller support also perform well. Genres that contain ambiguous events such as the *event* and *miscellaneous* genres do not perform well. This is expected as the events in these genres can be very different.

An F1-score of 0.8110 is significantly higher than chance level. However, if the automatic classification was used on an operating website, the website team would have to keep checking and adjusting the

genres. A possible improvement to this problem would be a bigger dataset. Halfway through the project, approximately 500 new events were added to the database. The average F1-score increased by 0.07 which is a significant improvement. The largest genre, *music*, has the highest f1-score of 0.895. If all genres had the same support, the average F1-score could grow closer to an amount that is acceptable for an operating website like Ugenda.

The sub-questions give an answer to the main research question and are answered as follows:

- *Are title and event description enough for a satisfactory performance or are other event details needed?*
  No, other features such as location are important. However, this still does not yield a result that is good enough for an operating website.
- *What methods of data preparation are most useful?*
  For this dataset, adding a location feature has the most significant performance improvement. However, a combination of natural language processing and adding date, time and location features performs good as well. Natural language processing may be more suitable for a larger dataset.
- *Which classification techniques are most useful and how do they perform compared to each other?*
  For this dataset and this multi-label classification problem, the logistic regression classifier performs the best. The random forest and support vector machine classifier have similar results to each other and the Naïve Bayes classifier performs the worst.

6.2 FUTURE WORK

6.2.1 *N-grams*

The n-gram approach mentioned in Section 2.2.1 could detect two-word jargon of specific genres. It could be interesting to see how this effects the performance of the classifiers.

6.2.2 *Dataset imbalance*

The only thing that was implemented to account for the imbalance was an adjusted performance metric. It could be interesting to see if one of the methods mentioned in Section 2.3.2 could improve performance.

6.2.3 *Toleration for similar category misclassification*

Some wrong predictions are not as wrong as others. Classifying a *kids* event to the *party* genre is a bigger mistake than classifying a *music* event to the *party* genre. We could determine which misclassifications can be tolerated, which could improve performance.

BIBLIOGRAPHY

[1] M. Ikonomakis, S. Kotsiantis, and V. Tampakas. "Text classification using machine learning techniques." In: *WSEAS Transactions on Computers* 4.8 (2005), pp. 966–974.

[2] A. K. Uysal and S. Gunal. "The impact of preprocessing on text classification." In: *Information Processing & Management* 50.1 (2014), pp. 104–112.

[3] D. Sculley and G. M. Wachman. "Relaxed online SVMs for spam filtering." In: (2007), pp. 415–422.

[4] M. Kandias, V. Stavrou, N. Bozovic, and D. Gritzalis. "Proactive insider threat detection through social media: The YouTube case." In: (2013), pp. 261–266.

[5] T. Abou-Assaleh, N. Cercone, V. Keselj, and R. Sweidan. "N-gram-based detection of new malicious code." In: 2 (2004), pp. 41–42.

[6] S. E. Robertson and K. S. Jones. "Relevance weighting of search terms." In: *Journal of the American Society for Information science* 27.3 (1976), pp. 129–146.

[7] Y. Yang and J. O. Pedersen. "A comparative study on feature selection in text categorization." In: 97 (1997), pp. 412–420.

[8] R. Kohavi. "A study of cross-validation and bootstrap for accuracy estimation and model selection." In: 14.2 (1995), pp. 1137–1145.

[9] I. S. Dhillon. "Co-clustering documents and words using bipartite spectral graph partitioning." In: (2001), pp. 269–274.

[10] F. Sebastiani. "Machine learning in automated text categorization." In: *ACM computing surveys (CSUR)* 34.1 (2002), pp. 1–47.

[11] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, and B. Scholkopf. "Support vector machines." In: *IEEE Intelligent Systems and their Applications* 13.4 (1998), pp. 18–28.

[12] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[13] A. van den Bosch, B. Busser, S. Canisius, and W. Daelemans. "An efficient memory-based morphosyntactic tagger and parser for Dutch." In: *LOT Occasional Series* 7 (2007), pp. 191–206.

[14] S. van der Walt, S. C. Colbert, and G. Varoquaux. "The NumPy array: a structure for efficient numerical computation." In: *CoRR* abs/1102.1523 (2011). URL: http://arxiv.org/abs/1102.1523.

[15] E. Jones, T. Oliphant, and P. Peterson. "SciPy: Open source scientific tools for Python." In: (2001–). [Online; accessed 2016-06-21]. URL: http://www.scipy.org/.

[16] F. Van Eynde. "Part of speech tagging en lemmatisering van het Corpus Gesproken Nederlands." In: *KU Leuven* (2004).