VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
DEPARTMENT OF COMPUTATIONAL AND DATA MODELING

Bachelour Thesis

Website Classification Using Machine Learning Approaches

Done by:
Domantas Meidus signature

Supervisor:
Assoc. Prof., Dr. Linas Bukauskas

Vilnius
2019

Contents

Abstract

World Wide Web (WEB) become one of the biggest information sources of websites with different contents. Websites content could be classified into categories - in that way all websites would be more structurized and indexed for people who are searching particular information. In this paper, websites classification is implemented by using Machine Learning and Natural Language Processing methods which allows to create a models with ability to classify website according to website content data. Website classification models are based on the most frequent words in each website category and each website set of content words which are transformed into Machine Learning features by using Natural Language Processing approach. Models capabilities of classifying websites was tested on the World Wide Web (WEB) english content websites and the best model is capable to predict correctly category with ~ 70 % accuracy.

Key words: *Machine Learning, Natural Language Processing, Web scrapping, WEB classifying, Data Preprocessing, Statistics*

Santrauka

Internetinių puslapių klasifikacija naudojantis save mokinančios kompiuterinės sistemos metodais

Pasaulinis tinklas (WEB) tapo viena iš didžiausių internetinių puslapių šaltinių saugykla. Puslapių turinys gali būti klasifikuojamas į kategorijas - tokiu būdu internetiniams puslapiams yra sukuriama struktūra ir juos lengviau indeksuoti, dėl to žmonėms surasti informaciją pasauliniame tinkle yra lengviau. Šiame straipsnyje, internetinių puslapių klasifikavimas yra atliekamas naudojantis save mokinančios kompiuterinės sistemos ir natūralios kalbos apdorojimo metodais, kuriais naudojantis yra sukuriami modeliai su internetinių puslapių klasifikavimo funkcionalumu, naudojantis internetinių puslapių turiniu. Internetinių puslapių klasifikavimo modeliai yra sukurti remiantis populiauriausių žodžių aibėmis kiekvienai internetinio puslapio kategorijai ir kiekvieno internetinio puslapio turinio žodžių aibėmis, kurios yra transformuojamos į save mokinančios kompiuterinės sistemos savybių aibę, naudojantis natūralios kalbos apdorojimo metodais. Modelių sugebėjimai klasifkuojant internetines svetaines buvo išbandomi naudojantis pasaulinio tinklo (WEB) anglų kalbos turinio internetinėmis svetainėmis. Geriausias sukurtas modelis geba klasifikuoti internetinius puslapius $\sim$ 70 % tikslumu.

Raktiniai žodžiai: *Save mokinačios kompiuterinės sistemos, Natūralios kalbos apdorojimas, Automatinis informacijos surinkimas, WEB klasifikavimas, Duomenų apdorojimas, Statistika*

Introduction

Website classification is a well known problem in computer science field. World Wide Web (WEB) consist of a lot of websites with a di erent categories. The motivation of classified websites would be significant - that would allow index websites categories by their content so it would be easier for people to find information among many websites.

The article is oriented to the two big topics in the data science: Machine Learning and Natural Language Processing. Article consist of three main parts - first two topics is theory based and the third topic is an implementation part:

1. **Natural Language Processing (NLP)**

   Natural Language Processing topic is oriented to introduce fundamentals of natural language processing methods. In this section is explained NLP types, problems and methods that are used in the implementation part.

2. **Machine Learning (ML)**

   Machine Learning topic is oriented to introduce fundamentals of machine learning theory. Since machine learning by itself is a big field of research, the structure of this section is divided into three parts:

   (a) Supervised Learning.

   (b) Unsupervised Learning.

   (c) Machine Learning models evaluation.

3. **Machine Learning and Natural Language Processing methods implementation in practise**

   In implementation part there is explained of how data sets are preprocessed using NLP methods and how models are trained to be capable of classifying real world websites. Methods used in the implementation part by creating models are covered in theories part so all sections are related to each other.

The main goal of this article is to explain all process for creating models which are capable of predicting websites categories based on websites content features.

Results of this project: created machine learning and custom models that are capable of classifying english content websites. Models performances are tested on a di erent websites data sets which two main factors of each data set is Website URL with website category. Machine Learning models performance evaluation on original data set was used Accuracy, Precision, Recall, F1 scores while on custom human made data set was, models performance was calculated by correctly and incorrectly predicted categories websites ratio.

1 Natural Language Processing approach

Natural language processing (NLP) is a automatic computational processing of human languages. It is computer science area of research that explores how computers can be used to understand and manipulate human language text or speech. One of the most frequent challenges in the NLP are:

1. **Speech recognition**
2. **Natural language understanding**
3. **Natural language generation**

In this report the Natural language understanding methods would be covered since the project implementation part is using NLP for analyzing and understanding given natural language data. One of the most di cult challenges for Natural language processing is to understand natural human language which is inherently ambiguous, and not well defined. The NLP requires more statistical algorithmic approach since basic logic, rules and ontologies methods are not fully enough to build a fully working model which could understand natural human language.

The classification problems in Natural language can be categorized in the several types:

1. **Word**. In "word" called problems which tries to answer to these pseudo questions: in what language the word is written? What other words are similar to the given word? Is the word misspelled or it is written correctly? How common the word is?

   These kind of problems are quite rare, as for many words interpretation depends on the context in which they are used

2. **Texts**. "Texts" problems usually faced with an input text such as a phrase, a sentence, a paragraph or a document. These kind of problems seeking to answer the questions such as: is it spam or not? It is about sports or economy? Is it ironic? Is it reliable? What kind of age group the text is created for? and so on.

   These types of problems are very common, and they are referred as document classification problems.

3. **Paired Texts**. The input of these problems are pair of words or longer texts. Usually these problems tries to answer the question about the paired words input such as: Are words A and B synonyms? Is word A a valid translation for word B? Are documents A and B written by the same author?

4. **Word in context** The input of these problems are piece of text and a particular word in it(phase, letter etc.). The main goal is to classify the word in the context of the text. The problem tries to cover these kind of questions: does the word or phrase refers to the person, location or organization? Is the word a noun, verb or an adjective?

5. **Relation between two words**. The input of these problems is two words or phrase within the context of a larger document and the main goal of the problem is to say the relations between words. Is word A the subject of vert B in a given sentence?

The natural language processing section would be covered with topics:

1. Words tokenization

2. Stop words
3. Term Frequency
4. N - grams words approach

1.1 Word Tokenization

Tokenization [6,] is the process of splitting a string into a list of pieces or tokens. Word tokenization is a method of breaking up a piece of text into many pieces, such as sentences and words. A token is a piece of a whole, so a word is a token in a sentence, and a sentence is a token in a paragraph. The main goal of words tokenization is to split the sentence into vector of words. In this way each word of the sentence could be analyzed and used in the further development of NLP methods.

There are multiple types of word tokenization:

1. **Tweet tokenizer**. Tweet tokenizer could be used for parsing tweets into tokens. The tweet term is well known in the Twitter platform. The tweet have two properties: the text characters length is limited and there are special tags which refers to the group or other person. The example of the tweets tokenization:

```
input = "This is a cooool #dummysmiley: :-) :-P <3 and some
    arrows < > -> <--"
```

   The tweet tokenizer output of the tokenized text:

```
['This', 'is', 'a', 'cooool', '#dummysmiley', ':', ':-)', ':-P',
     '<3', 'and', 'some', 'arrows', '<', '>', '->', '<--']
```

2. **Multi-Word Expression Tokenizer**. A Multi-Word Expression Tokenizer takes a string that has already been divided into tokens and retokenizes it, merging multi-word expressions into single tokens

3. **Regular-Expression Tokenizers**. A RegexpTokenizer splits a string into substrings using a regular expression. For example, the following tokenizer forms tokens out of alphabetic sequences, money expressions, and any other non-whitespace sequences:

```
input = "Good muffins cost $3.88\nin New York.  Please buy me\
    ntwo of them.\n\nThanks."
regular_expression = ('\w+|\$[\d\.]+|\S+')
```

   The regular exoression output of the tokenized text:

```
['Good', 'muffins', 'cost', '$3.88', 'in', 'New', 'York', '.',
'Please', 'buy', 'me', 'two', 'of', 'them', '.', 'Thanks', '.']
```

4. **S-Expression Tokenizer**. Is used to find parenthesized expressions in a string. In particular, it divides a string into a sequence of substrings that are either parenthesized expressions (including any nested parenthesized expressions), or other whitespace-separated tokens.

5. **Simple Tokenizers**. These tokenizers divide strings into substrings using the string split() method. When tokenizing using a particular delimiter string, use the string split() method directly, as this is more e cient.

Word tokenization requires an input of text, for instance:

```
input_text = "Tokenization is the act of breaking up a sequence of
    strings into pieces."
```

Figure 1. Input data

. This piece of text would be tokenized into vector as:

```
tokenized_text = ['Tokenization', 'is', 'the', 'act', 'of', 'breaking',
    'up', 'a', 'sequence', 'of', 'strings', 'into', 'pieces', '.']
```

Figure 2. Output after words tokenization process

In english language the tokenization process is quiet straightforward because usually text is splitted based on whitespace or punctuation. However in other languages like Hebrew or Arabic words tokenization could be more challenging because of language properties - some words attaches to the next one without whitespace.

1.2 Stop Words

Stop words in Natural language processing is a term that defines the most common words in a language which have no valuable use from it. Each language has a di erent set of "stop words" so there is no single universal list of stop words. Stop words belongs to the language set so it is cannot be removed by creating rules since they not have common pattern in a di erent languages. There are several ways of detecting stop words:

1. Use a generated list of all stop words in a particular language. There are plenty of open source stop words list for each language. This option is reliable when text is general and a task is to rid of well know common words which appears into particular language by the open source dataset creators. There are plenty of sources of stop word dataset list download.

2. Determine stop words of their frequency in a text. This method is applied for a text which may contain more stop words based on the problem where open source stop list datasets do not contain them. This method counts the frequency of words in the document. After that there are filtering methods which depends on the problem and text:

   (a) Filter out words which occurs more than x % where x is a integer number between [0,100]. x number value depends on the language and the problem. It is a good practise to eliminate more than 85% frequent words in the text in order to be e ective in several text mining tasks.

(b) Least frequent terms as stop words. Some words could be used few times in a text but they might be not proper for NLP tasks. These words could be tags, tweets from the social networks or made-up terms by people. These kind of words may occur rarely in the text so it could be tracked and eliminated.

(c) Calculating Inverse Document Frequency of each word. IDF value is quicly decreasing as word becomes more common in documents. Most stopwords, due to their prevalence, will have an IDF near one.

1.3 Term Frequency

Some problems of NLP requires calculate words frequency in order to get necessary results. This allows di erentiate between how many times each word is used. Calculate word count is not a simple task as it primary appears, because there are some flaws which should be though over before producing words counting operations:

1. The word cases should be normalized. Text in the virtual environment normally are produced from ASCII character table. Each symbol has an unique code and these codes are interpreted into human readable symbols. For this reason words like "Computer" and "computer" are not equal, because their symbols do not match although these words have the same meaning. This should be consider before counting words frequency and normalize text by transforming letters to uppercase or lowercase in the text.

2. Removing words stemmings. Some words could be written in the other tenses which may transform their form, for instance: english verbs like *announces, announced and announcing* - these words have been made of word announc. Normally these words in the word counting would be treated as di erent words but if their prefix would be removed then it possible to normalize these words to their "primal" form.

3. Removing stopwords.

Term Frequency [4] could be defined as an integer-valued feature for a word *w* as below:

$$\phi w(x,c) = \begin{cases} TFwc & \text{if } w \text{ is in both } x \text{ and } c, \\ 0 & \text{othervise.} \end{cases} \tag{1.1}$$

c is a document

w is a word

$TFwc$ is the number of times the word w appears in document c

1.3.1 Inverse Document Frequency

Inverse Document Frequency(IDF) [10]. IDF is a weight indicating how widely a word is used in the documents. The more frequent the words is used the lower score it gets. IDF is calculated by formula:

$$IDF(w) = 1 + \log_{\frac{N}{Nw}} \tag{1.2}$$

N Total number of documents

w is a word

Nw Number of documents containing word w

Term frequency measure of how prevalent a term is in a single document. The Inverse Document Frequency is used when calculating of how common is a word in the more than one document. IDF method could reveal rare words that could be not necessary and can be removed from the words list. When a term is very rare the IDF tends to be at the higher value and in opposite - when a term is more common in documents, the IDF value decreases.

1.3.2 Term Frequency Inverse Document Frequency (TFIDF)

Term Frequency Inverse Document Frequency (TFIDF) [6] is a representation for text of Term Frequency (TF) and Inverse Document Frequency (DF). The TFIDF value of term *t* in a given document *d* is thus:

$$TFIDF(t, d) = TF(t, d) \times IDF(t) \tag{1.3}$$

d is a single document

t is a word

TFIDF value is specific to a single document (d) whereas IDF depends of the entire documents. Each document thus becomes a feature vector, and the documents is the set of these feature vectors.

1.4 N - grams words approach

N-grams [4,] are all combinations of adjacent words or letters of length N that could be found in the source text. N-grams method capture the language structure from the statistical point of view, like what letter or word is likely to follow the given one. The longer n-gram(the higher n), the more text should be applied in constructing pairs of n words. For example, pairs of adjacent words for a sentence "The quick brown fox jumps" would be transformed into the set of its constitutent words quick, brown, fox, jumps, plus the tokens: quick_brown, brown_fox and fox_jumps. This representation of words is called n-grams. Adjancent pairs are commonly called bi-grams.

N-grams are useful when the words in pairs have more meaning than words individually. For example: new_york_building 3-gram(in other words it could be named as trigram) would be more informative and would have di erent meaning than each word individually: new, york and building.

The advantage of using n-grams approach that it is not requiring linguistic knowledge or complex parsing algorithm in order to generate one.

The disadvantage of n-grams is that they greatly increase the size of the feature set. There are many adjacent words pairs, and still more adjacent word triples.

Example of N-grams approach:

Sample sequence	1-gram sequence	2-gram sequence	3-gram sequence
	Unigram	Bigram	Trigram
to be or not to be	to, be, or, not, to, be	to be, be or, or not, not to, to be	to be or, be or not, or not to, not to be

Table 1. Example of n-gram sequences

2 Machine learning approach

Machine learning (ML) is the study of algorithms and mathematical models with a feature of progressively improve a performance on a specific task. Machine learning is a subfield of Artificial Intelligence (AI). The main di erence between Artificial Intelligence and Machine learning is that machine learning performance results primary depends on the data set and it makes data driven decisions while Artificial Intelligence involves agents at the top. Another subfield of machine learning is Deep Learning that uses a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Hierarchy representation of Artificial Intelligence, Machine Learning and Deep Learning could be seen in the 3 figure.

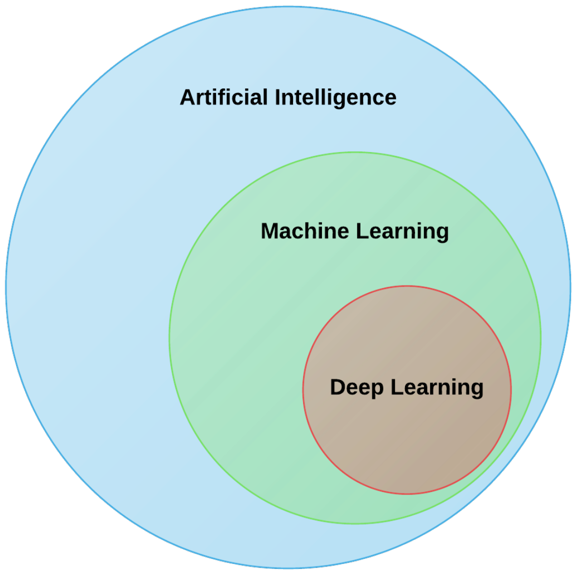


Figure 3. Hierarchy representation of AI, ML and Deep Learning

Source: https://www.codesofinterest.com/2016/11/di erence-artificial-intelligence-machine-learning-deep-learning.html

Machine learning models is divided into two types:

1. **Supervised learning** - is the machine learning task of learning a function that maps an input to an output based on example input-output pairs.
2. **Unsupervised learning** - is the machine learning tasks that learns from test data that has not been labeled, classified or categorized.

2.1 Supervised Learning

Supervised Machine learning models are all about finding appropriate representations for their input data and it requires 3 things:

1. **Input data**. A good data set is the key of creating good Machine Learning model. The input data depends on the problem which the developer want to solve - if the task is speech recognition, then the input data could be sound files converted to the form that computer could proceed, for example: binary. If the task is image tagging, then the data could be a

pictures where each pixel is converted to the RGB (red-green-blue) format or HSV (hue-saturation-value) format. The input data depends on the problem and the final goal of the problem.

2. **Output data**. Output data in other words could be called as results of the input data. Supervised machine learning models should know the results of each data input entry point in order to find out a pattern which helps to predict a results.

3. **Validation**. Validation is a way to measure whether the algorithm is doing a good job to determine the distance between the algorithm's current output and its expected output. The validation for supervised machine learning models is split into 2 parts: training data and testing data. Training data is input data and output data which is used for machine learning model training and it from where model learns the patterns of the data which produce certain output. The testing data is used after the machine learning training and from this data the model is evaluated of how successfully it predicted the output results from the input data.

Supervised learning is grouped into two types:

1. **Classification** A classification type is when the output variable is representing a category

2. **Regression** A regression type is when the output variable is representing a real value

2.1.1 Classification

Classification is the process of predicting the class of given data points. Classes are usually named as a term of **Labels**. Classification main goal is from given input data points predict an output which would mark a label. Classification predictive modeling: approximating a mapping function (f) from input variables (X) to discrete output variable (y).

Classification Machine Learning algorithms:

1. **Linear Classifiers** is the statistical classification group of identifying classes of the object's characteristics. A linear classifier methods makes a classification decisions based on the values of a linear combinations of the characteristics. In this paper two linear classifiers would be described: *Logistic Regression* and *Naive Bayes Classifier*

   (a) **Logistic Regression** Logistic Regression is a statistical method for analysing a data set in which there are one or more independent variables that determine an measured(outcome) with a dichotomous variable. It predicts the probability of an outcome that have two values.

   The main goal of logistic regression is to find the best fitting model to describe relationship between the dichotomous characteristics of interest and a set of independent variables.

   Logistic regression generates a logistic curve (yellow color) (the Linear and logistic model representation figure 4) which y-axis is limited to values between 0 and 1; [0, 1].

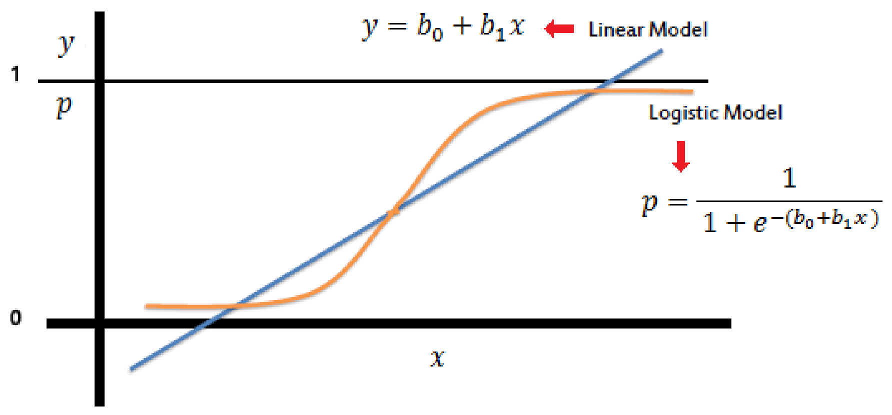


Figure 4. Linear model and Logistic model representation [**?**]

Logistic model mathematical representation: $p = \frac{1}{1+e^{-(b_0+b_1x)}}$
Linear model mathematical representation: $y = b_0 + b_1 x$
Logistic regression could be expressed in the formula:

$$\frac{p}{1-p} = \exp b_0 + b_1 x \tag{2.1}$$

p Logistic Model
b_0 Logistic regression constant
b_1 The slope that defines the steepness of the curve

(b) **Naive Bayes Classifier**.
Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Bayes theorem could be expressed in the mathematical equation:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.2}$$

A **and** B are events
$P(B) \neq 0$
$P(A|B)$ is a conditional probability: the likelihood of event A occurring given that B is true.
$P(B|A$ is also a conditional probability: the likelihood of event B occurring given that A is true.
$P(A)$ **and** $P(B)$ are the probabilities of observing A and B independently of each other; this is known as the marginal probability

Naive Bayes [11] is e ective in many practical applications including text classification, performance management and medical diagnosis. The e ectiveness of Naive Bayes classifier comes from it is presence of feature dependencies: optimality in terms of zero-one loss (classification error) is not necessarily related to the quality of the fit to a probability distribution.

2. **Support Vector Machines(SVM)**

   Support vector machine(SVM) is a discriminative classifier defined by a separating hyperplane. Hyperplane is a line dividing a plane i two parts where in two parts. SVM algorithm main objective is to find a optimal hyperplane in an N-dimensional (N - is the number of features) space that distinctly classifies the data points.

   When using Support Vector Machine it should be considered these parameters:

   - Input. Set of training pair samples. Call the input sample features $x_1, x_2 ... x_n$ and the output result y
   - Output. Set of weights w. One for each feature, whose linear combination predicts the value of y

   SVM usage in the real world applications:

   - Text and Hypertext categorization. SVM method could significantly reduce the need for labeled training instances in both the standard inductive and transductive settings.
   - Classification of images. SVM achieve higher search accuracy than traditional query refinement schemes
   - Biological and other sciences. SVM performed good results in classification of proteins schemes or classifying permutation tests.

3. **Decision Trees**

   Decision tree are flowchart-like structures that classifies input data points or predict output values given inputs. A decision tree is a decision-making device which assigns a probability to each of the possible choices based on the context of the decision: $P(f|h)$, where f is an element of the set of choices and h is the context of the decision. Probability $P(f|h)$ is determined by asking the sequence of questions $q_1, q_2, ..., q_n$ about the context, where the ith question asked is uniquely determined by the answers to the $i - 1$ previous questions [9].

   Decision tree builds classification or regression models in the form of a tree data structure. The data sets is break into smaller subsets. Decisions are represented as nodes and leaf nodes in the tree.

   A decision tree consist of three types of nodes:

   (a) Decision nodes - represented by squares
   (b) Chance nodes - represented by circles
   (c) End nodes - represented by triangles

   Advantages of Decision trees classification:

   - Easy to understand and interpret because of the tree structure.
   - The results could be achieved even with the little data.
   - Could determine best, worst and expected values for di erent scenarios

   Disadvantages of Decision trees classification:

- It could be unstable. Small changes could imply large changes in the structure of the decision tree
- They are often relatively inaccurate comparing with others classification algorithms
- Information gain in decision trees is biased in favor of those attributes with more levels

2.1.2 Regression

Another type of Machine learning is called Regression. Regression algorithms tries predict a value for an input based on previous information. The main di erence of classification type and regression type is that regression main goal is to estimate a value while classification type main goal is to estimate a class of an observation.

Regression models have the following parameters and variables:

- **The unknown parameters**, denotes as β, which may represent a scalar or a vector
- **The dependent variable, Y**. This is a main factor that has to be understood and predicted.
- **The independent variables, X**. This is a factor which have an impact on dependent variable

A regression model relates Y to a function of X and β: $Y \approx f(X, \beta)$

Regression Machine Learning algorithms:

1. **Linear Regression**

   Linear regression [8] is a technique that analyze the relationships between variables and how they contribute and related to producing a particular outcome. Linear regression establishes a relationship between **dependent variable (Y)** and **independent variables (X)** using straight line also known as regression line.

   Linear regression is represented in the formula:

$$Y = a + b \times X + e \tag{2.3}$$

   Y Dependent variable

   X Independent variables

   a Intercept

   b Slope

   e Error term

   This equation represents of how Linear Regression method predicts value of target variable on given predictor variables.

   Example of the linear regression scatter plot could be seen in the figure 5. The black line consists of the predictions, the points are the data and the vertical lines between the points and the black line represent errors of prediction.

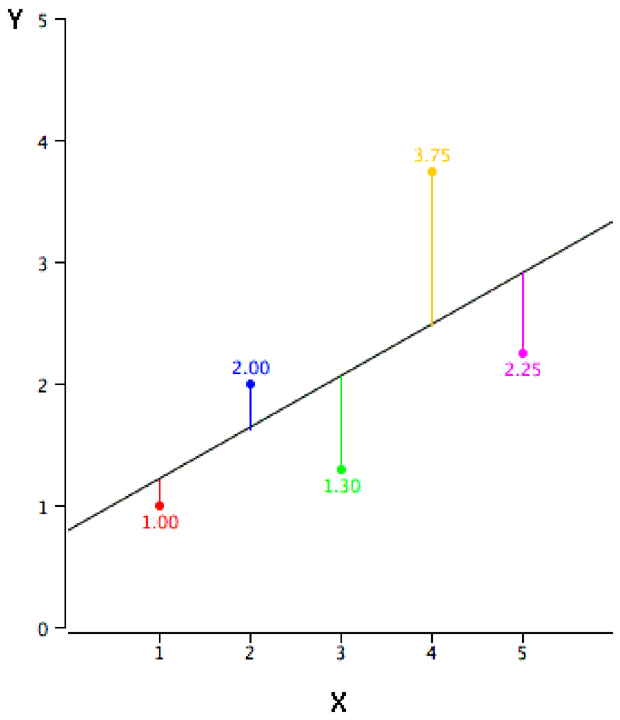


Figure 5. Scatter plot of the linear regression method [1]

Logistic regression properties:

- Linear regression method requires linear relationship between independent variable (Y) and dependent variables X
- Linear regression results could be drastically dependent on Outliers. Outliers are the Data points that diverge in a big way from the overall pattern.
- Multicollinearity could increase the variance of the coe cient estimates and make the estimates sensitive to minor changes in the model. Multicollinearity is a statistical process in which predictor variables are correlated.

Advantages of using logistic regression model:

- Space complexity of the logistic regression model is low because it needs only to save the weights at the end of training
- Good interpretability, simple to understand
- Feature importance is generated at the time model building. Dimensionality reduction could be achieved by handling feature selection and by using hyperparameters

Disadvantages of using logistic regression model:

- Multicollinearity should be avoided
- Prone to outliers
- Linear regression assumes that the data is independent

2. **Polynomial Regression**

   Polynomial Regression method is the relationship between the **independent variables X** and the **dependent variable Y** which is modelled as an nth degree polynomial. This method is good for handling non-linearly separable data. This method fits a nonlinear relationship between the value of x and the corresponding conditional mean of Y, denoted as $E(Y|X)$.

   nth degree polynomial regression method could be described in the mathematical formula:

   $$Y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + ... + \beta_n x^n + e \tag{2.4}$$

   Y Dependent variable

   x Independent variables

   β Unknown parameters

   e Error term

   Polynomial regression properties:

   - Otherwise than linear regression, polynomial regression method could model non-linearly seperable data
   - Full control over the modelling of feature variables
   - Prone to over fitting if exponents are not selected according to model design.

   Polynomial regression have few problems that should be taken to consideration before using this algorithm:

   (a) When the X values are large, the model could be leaded to the math overflow which could make results inaccurate.

   (b) The parameters of the model are intertwined so that lead=s of having covariance and dependency cases between the X values.

3. **Ridge Regression**

   Ridge regression is a remedial measure taken to alleviate multicollinearity amongst regression predictor variables in a model. By adding a degree of bias to the regression estimates, ridge regression reduces the standart errors rate.

   Ridge regression model formula:

   $$Y = X\beta + e \tag{2.5}$$

   Y Dependent variable

   X Independent variables

   β Regression coe   cients to be estimated

   e Error term

   The presence of multicollinearity could be detected in a few ways:

   - A regression coe   cient is not significant even though variables should be correlated with dependent variable Y

- The regression coe cients changes dramatically if X independent variables are added or deleted
- X independent variables have high pairwise correlations

Features of ridge regression model:

- The assumption is the same as least squared regression
- The value of coe cients does not reach zero

4. **Lasso Regression**

   Lasso regression [5] method performs both variable selection and regularization in order to enhance the prediction. Lasso regression and ridge regression are similar methods but lasso regression is using an absolute value bias while ridge regression method is using squared value bias.

   The goal of lasso method is to obtain the subset of predictors that minimizes prediction error for a quantitative response variable. This is done by shrinking variables towards zero.

   The lasso estimate is defined by formula:

   $\beta^{lasso} = argmin \sum_{i=1}^{N}(y_i - \beta 0 - \sum_{j=1}^{P} x_{ij}\beta_j)^2$ subject to $\sum_{j=1}^{P} |\beta_j| <= t$

   Lasso regression translates each coe cient by a constant factor λ, truncating at zero. This process is called "soft thresholding," and is used in the context of wavelet-based smoothing.

   Lasso regression is often an e ective technique for shrinkage and feature selection.

2.2 Unsupervised Learning

Unsupervised machine learning clusters only need to have an input data which would be used for detecting a patterns of the data.

2.2.1 Clustering

Clustering is a task of grouping a set of objects by their data property patterns. Clustering methods are used for identifying similar groups to entities of that group than those of the other groups. Clustering methods are diverse by their types:

1. **Centroid-based**

   Clustering model which is related to the notion of similarity is derived by the closeness of a data point to the centroid of the clusters. Using algorithms with this type it is important to know the number of clusters before grouping data.

2. **Distributed-based**

   Clustering model which is based on the notion of how probable is it that all data points in the cluster belong to the same distribution. Distributed-based models tends more su er from overfitting.

3. **Connectivity-based**

   Clustering model which is based on the notion that the data points closer in data space exhibit more similarity to each other than the data points lying farther away. This model could be diverse into two approaches:

(a) Classifying all data points into separate clusters and then aggregating them as the distance decrease.

(b) All data points are classified as a single cluster and then partitioned as the distance increases.

Models are very easy to interpret but lacks scalability for handling big data sets

4. **Density-based**

   Clustering model which is search the data space for areas of varied density of data points in the data space. It isolates various di erent density regions and assign the data points within these regions in the same cluster.

1. **K-Means Clustering**

   K-mean is a centroid model based clustering algorithm which the main goal is to partition the inputs into sets $S1, ..., S_k$ in a way that minimizes the total sum of squared distances from each point to the mean of its assigned cluster. k representing the number of clusters.

   K-means [7] clustering abstract algorithm:

   (a) Decide on a value for K, the number of clusters

   (b) Initialize the K cluster centers

   (c) Decide the class memberships of the N objects by assigning them to the nearest cluster center

   (d) Re-estimate the K cluster centers, by assuming the memberships found above are correct

   (e) Repeat c and d until none of the N objects changed membership in the last iteration

   Advantages of K-mean clustering :

   - K-mean clustering is fast. Computing the distances between points and group centers requires few computations. The linear complexity of the algorithm is **O(n)**
   - An instance could move to another cluster when the centroids are recomputed
   - Converges to local minimum of within-cluster squared error

   Disadvantages of K-mean clustering:

   - The exact number of clustering groups/classes should be known before using k-means clustering algorithm
   - K-means starts with a random choice of cluster centers and it could perform di erent results for each algorithm run
   - The order of the data has an impact to the final results
   - Sensitive to outliers
   - Detects spherical clusters only

2. **Mean-Shift Clustering**

   Mean-shift [2] clustering algorithm is a non parametric clustering technique which does not require prior knowledge of the number of clusters and does not constrain the shape of the clusters.

$$K(x) = \left\{ \begin{array}{c} 1 \text{ if } ||x|| <= \lambda \\ 0 \text{ if } ||x|| > \lambda \end{array} \right\} \tag{2.6}$$

Let data be a finite set S embedded in the n-dimensional Euclidean space, X. Let K be a flat kernel that is the characteristic function of the λ-ball in X

The sample mean at x $\in Xis$:

$$m(x) = \frac{\Sigma_{s\in S} K(s-x)s}{\Sigma_{s\in S} K(s-x)} \tag{2.7}$$

The di erence $m(x) - x$ is called *mean shift*. In each iteration of the algorithm, $s \Leftarrow m(s)$ is performed for all $s \in S$ simultaneously. The mean shift vector always points toward the direction of the maximum increase in the density.

Mean-shift clustering abstract algorithm:

(a) Circular sliding window centered at a randomly selected point C and having radius r as the kernel.

(b) At every iteration the sliding window is shifted towards regions of higher density by shifting the center points to the mean of the points withing the window while there is direction at which a shift can accommodate more points inside the kernel

(c) This process of steps (a) and (b) is done with many sliding windows until all points lie within a window.

Advantages of Mean-Shift clustering:

- Model free - does not assume any prior shape on data clusters
- It requires single parameter of window size h
- Robust to outliers

Disadvantages of Mean-Shift clustering:

- Results depends on window size
- Window size selection is not trivial
- Computationally expensive
- Does not scale well with dimension of feature space

3. **Density-Based Spatial Clustering of Applications with Noise (DBSCAN)**

   Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a density based clustering algorithm. Given a set of points in some space, it groups together points that are closely packed together, marking as outliers points that lie alone in low-density regions. Overall average complexity of DBSCAN clusetering algorithm is **O(log n)**.

   DBSCAN abstract algorithm:

   (a) Find the points in the ϵ neighborhood of every point, and identify the core points with more than $minPts$ neighbors
   (b) Find the connected components of core points on the neighbor graph, ignoring all non-core points
   (c) Assign each non-core point to a nearby cluster if the cluster is an ϵ neighbor, otherwise assign it to noise

   Parameters of DBSCAN clustering algorithm:

   - $MinPoints$: The minimum number of points to form a dense region
   - ϵ: The minimum distance between two points. The points are considered neighbors if the distance between two points is lower or equal to ϵ value

   Advantages of DBSCAN clustering algorithm:

   - DBSCAN does not require one to specify the number of clusters in the data
   - DBSCAN can find arbitrarily shaped clusters
   - DBSCAN has a notion of noise, and is robust to outliers
   - DBSCAN requires just two parameters and is mostly insensitive to the ordering of the points in the database
   - DBSCAN is designed for use with databases that can accelerate region queries, e.g. using an R* tree

   Disadvantages of DBSCAN clustering algorithm:

   - DBSCAN is not entirely deterministic: border points that are reachable from more than one cluster can be part of either cluster, depending on the order the data are processed
   - The results of DBSCAN model depends on the distance measure
   - DBSCAN cannot cluster data sets well with large di erences in densities
   - Choosing a meaningfull distance treshhold could be di cult if the data is not proper for this algorithm

4. **Expectation–Maximization (EM) Clustering**

   The Expectation–Maximization clustering algorithm is an iterative method to find maximum likelihood or maximum a posteriori (MAP) which computes probabilities of cluster memberships based on one or more probability distributions. The goal of the EM clustering algorithm is to maximize the overall probability or likelihood of the data.

   The EM algorithm seeks to find the MLE of the marginal likelihood by iteratively applying these two steps:

(a) Expectation step (E step): Define $\Theta(\theta|\theta^{(t)})$ as the expected value of the log likelihood function of $theta$, with respect to the current conditional distribution of Z given X and the current estimates of the parameters $\theta^{(t)}$:

$$\Theta(\theta|\theta^{(t)}) = E_{Z|X_1\theta^{(t)}}[\log L(\Theta; X; Z)] \tag{2.8}$$

(b) Maximization step (M step): Find the parameters that maximize this quantity:

$$\theta^{(t+1)} = argmax_\theta \Theta(\theta|\theta^{(t)} \tag{2.9}$$

X Observed data

Z a set of unobserved latent data or missing values Z

θ a vector of unknown parameters

$L(\theta, X, Z)$ likelihood function

Expectation–Maximization abstract algorithm:

(a) Initialize the parameters θ to some random values

(b) Compute the probability of each possible value of Z, given θ

(c) Use the just-computed values of Z to compute a better estimate for the parameters θ

(d) Iterate steps (b) and (c) until convergence

Advantages of Expectation–Maximization clustering algorithm:

- Likelihood is guaranteed to increase for each iteration
- Is a derivative-free optimizer
- Is fast if analytical expressions for the M-step are available
- Parameter constraints are often dealt with implicitly

Disadvantages of Expectation–Maximization clustering algorithm:

- Requires both forward and backward probabilities
- Significant implementational e ort required compared to numerical optimization
- I Convergence may be slow if analytical expressions for the M-step are not available since numerical optimization must be applied
- Hessian must be calculated manually

5. **Hierarchical Clustering**

Hierarchical clustering [3] is a method of cluster analysis which seeks to build a hierarchy of clusters. Hierarchical clustering algorithm run once and create a dendrogram which is a tree structure containing a k-block set partition for each value of k between 1 and n, where n is the number of data points to cluster allowing the user to choose a particular clustering.

Hierarchical clustering consist of two types:

(a) **Agglomerative**: This is a "bottom-up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. Bottom-up algorithms treat each data point as a single cluster at the outset and then successively merge pairs of clusters until all clusters have been merged into a single cluster that contains all data points. The time complexity of this method is $O(n^3)$.

(b) **Divisive**: This is a "top-down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy. Divisive clustering complexity is $O(2^n)$.

Agglomerative Hierarchical clustering abstract algorithm [12]:

(a) Compute the similarity between all the pairs of clusters

(b) Combine the foremost similar two clusters

(c) Update the similarity matrix to replicate the pairwise similarity between the new cluster and the original clusters

(d) Repeat steps b and c until only a single cluster remains

Advantages of Hierarchical clustering algorithms:

- Hierarchical clustering does not require us to specify the number of clusters
- Algorithm is not sensitive to the choice of distance metric

Disadvantages of Hierarchical clustering algorithms:

- Sensitivity to noise and outliers
- Breaking large clusters
- Di culty handling di erent sized clusters and convex shapes

2.2.2 Dimensionality reduction

Dimensionality reduction [14] is the transformation of high-dimensional data into a meaningful representation of reduced imensionality.Ideally, the reduced representation should have a dimensionality that corresponds to the intrinsic dimensionality of the data. The intrinsic dimensionality of data is the minimum number of parameters needed to account for the observed properties of the data.

The problem of dimeansionality reduction could be defined as follows: Data set is represented in a n x D matrix X consisting of n datavectors $x_i(i \in 1, 2, ..., n)$) with dimensionality D. The data set has intrinsic dimensionality $d(where d < D)$, in mathematical terms, intrinsic dimensionality means that the points in data set X are lying on or near a manifold with dimensionality d that is embedded in the D-dimensional space. Dimensionality reduction techniques transform dataset X with dimensionality D into a new data set Y with dimensionality d, while retaining the geometry of the data as much as possible. High-dimensional data point is denoted by x_i , where x_i is the ith row of the D-dimensional data matrix X. The low-dimensional counterpart of x_i is denoted by y_i , where y_i is the ith row of the d-dimensional data matrix Y. The data set X is assumed as a zero-mean.

There are two components of dimensionality reduction:

1. **Feature selection**. The subset of the original set of variables are found to get a smaller subset which could be used to model the problem. This component is divided into three separate parts:

   (a) *Filter*. Filter out features with small potential to predict outputs. Filter operation could be expressed in mathematical representation:

      i. Let Φ be a current set of features

      ii. Removing feature $\phi k(x)$ is possible only when:

$$\tilde{P}(y|\Phi|\phi_k) \approx \tilde{P}(y|\Phi) \tag{2.10}$$

      For all values of ϕ_k, y

   (b) *Wrapper*. Select features that directly optimize the accuracy of the classifier

   (c) *Embedded*. Features are selected to add or be removed while building the model based on the prediction errors

2. **Feature extraction**. Reduces the data in a high dimensional space to a lower dimension space

Advantages of Dimensionality Reduction:

- Improves performance in data compression
- Reduces computation time
- Removes redundant features
- Hence reduced storage space

Disadvantages of Dimensionality Reduction:

- It may lead to some amount of data loss.

1. **Principal Component Analysis (PCA)**

   Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. PCA is a linear transformation of d dimensional input x to M dimensional feature vector z such that under which the retained variance is maximal. It is also considered as the linear projection for which the sum of squares reconstruction cost is minimized.

   Goals of Principal Component Analysis (PCA):

   - Simplification
   - Data reduction
   - Outlier Detection
   - Variable selection
   - Classification
   - Prediction

- Unmixing

Many of the goals of PCA are concerned with finding relationships between objects. PCA estimates the correlation structure of the variables. The importance of a variable in a PC model is indicated by the size of its residual variance.

PCA [15] in matrix form is the least squares model:

$$X = 1\tilde{x} + TP' + E \tag{2.11}$$

$\tilde{x}$ Mean vector which is included in the model formulation

P' Projection matrix. It is also called loading vector

T Object coordinates in the plane. It is also called scoring vectors

X The projection

E The deviations between projections and the original coordinates are termed the residuals.

2. **Linear Discriminant Analysis (LDA)**

   Linear Discriminant Analysis (LDA) [16] is a method to find a linear combination of features that characterizes or separates two or more classes of objects or events.

   Given a data matrix $A \in^{N\times n}$, classical LDA aims to find a transformation $G \in^{N\times t}$ that maps each column a_i of A, for $1 \leq i \leq n$, in the N-dimensional space to a vector b_i in the ι-dimensional space. That is $G : a_i \in\Leftarrow b_i = G^T a_i \in^{\iota}$ $(\iota < N)$. Equivalently, classical LDA aims to find a vector space G spanned by $\mathrm{g}_{i i=1}$, where $G = [g_1, ..., g_\iota]$, such that each a_i is projected onto G by $(g_1^T a_i, ..., g^T a_i)T \in^{\iota}$.

   Assume that the original data in A is partitioned into k classes as $A = \Pi_1, ..., \Pi_k$, where Π_i contains n_i data points from the ith class, and $\sum i = 1^k n_i = n$. Classical LDA aims to find the optimal transformation G such that the class structure of the original high-dimensional space is preserved in the low-dimensional space. In general, if each class is tightly grouped, but well separated from the other classes, the quality of the cluster is considered to be high. In discriminant analysis, two scatter matrices, called within-class (S_w) and between-class (S_b) matrices, are defined to quantify the quality of the cluster, as follows [4]: $S_w = \sum i = 1^k \sum x \in \Pi_i (x - m_i)(x - m_i)T$, and $S_b = \sum i = 1^k n_i (m_i - m)(m_i - m)T$, where $m_i = \frac{1}{n_i} \sum x \in \Pi_i x$ is the mean of the ith class, and $m = \frac{1}{n} \sum i = 1^k \sum x \in \Pi | ix$ is the global mean.

   Notation:

   n number of instances in the data set

   k number of classes in the data set

   A_i ith instance in matrix representation

   a_i ith instance in vectors representation

   r r number of rows in A_i

   c number of columns in A_i

   N dimension of $a_i (N = r * c)$

   Π jth class in the data set

L transformation matrix (left) by Two-Dimensional Linear Discriminant Analysis

R transformation matrix (right) by Two-Dimensional Linear Discriminant Analysis

I number of iterations in Two-Dimensional Linear Discriminant Analysis

B_i reduced representation of A_i by Two-Dimensional Linear Discriminant Analysis

ι_1 number of rows in B_i

ι_2 number of columns in B_i

The goal of Linear Discriminant Analysis (LDA) is to project a data set onto a lower-dimensional space with good class-separability in order avoid overfitting and also reduce computational costs.

Pseudo algorithm for using LDA approach:

(a) Compute the d-dimensional mean vectors for the di erent classes from the data set

(b) Compute the scatter matrices

(c) Compute the eigenvectors $(e_1, e_2, ..., ed)$ and corresponding eigenvalues $(\lambda_1, \lambda_2, ..., \lambda_d)$ for the scatter matrices

(d) Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a $d \times k$ dimensional matrix W

(e) Use $d \times k$ eigenvector matrix to transform the samples onto the new subspace. This can be summarized by the matrix multiplication: $Y = X \times W$, where

X is a $n \times d$-dimensional matrix representing the n samples

Y are the transformed $n \times k$-dimensional samples in the new subspace

2.2.3 Association analysis

Association analysis is a method which is useful for discovering relationships hidden in large data sets. The uncovered relationships can be represented in the form of sets of items present in many transactions, which are known as **association rules** that represents relationships between two item sets. Association rule mining finds all rules in the database that satisfy some minimum support and minimum confidence constraints.

One of the real life examples in where association rules could be used is two items expression:
Olives $\rightarrow$ Wine

The rule suggest a relationship between olives usage with wine because olives is well know a good pair with wine.

An **association rule** [13] is an implication expression of the form $X \rightarrow Y$, where X and Y are disjoint item sets, i.e., $X \cap Y = \emptyset$. The strength of an association rule can be measured in terms of its support and confidence. Support determines how often a rule is applicable to a given data set, while confidence determines how frequently items in Y appear in transactions that contain X. These metrics has an formal definitions:

$$Support, s(x \rightarrow Y) = \frac{\delta(X \cup Y)}{N} \tag{2.12}$$

$$Confidence, c(x \rightarrow Y) = \frac{\delta(X \cup Y)}{\delta(X)} \tag{2.13}$$

2.3 Machine Learning models evaluation

Machine Learning models performance could be evaluated of how well they are classifying websites categories. Evaluation scores is a first level indicator which lets to know of Machine Learning capabilities of predicting categories according to the primary training features data sets.

Machine Learning models could predict website categories when training features set are fitted into model. After that model is fed with training features set and it outputs prediction set of predicted categories of websites. These predicted categories are compared with training labels set. Models predictions set and training labels set allows to generate evaluation scores and analysis of how well model is trained to predict categories of websites.

Calculating models predictions scores, there are 4 special terms which appears in the formulas:

- **True Positives (TP)** - is an outcome where the model correctly predicts the positive class. Example of true positive condition: *Person with disease was diagnosed a disease*.
- **True Negatives (TN)** - is a true negative is an outcome where the model correctly predicts the negative class. Example of true negative condition: *Person with no disease was not diagnosed a disease*.
- **False Positives (FP)** - is a result that indicates a given condition exists, when it does not. A false positive error is a type I error where the test is checking a single condition, and wrongly gives an a rmative decision. Example of false positive condition:: *Healthy person was diagnosed with a specific disease*.
- **False Negatives (FN)** - is a test result that indicates that a condition does not hold, while in fact it does. A false negative error is a type II error occurring in a test where a single condition is checked for and the result of the test is erroneously that the condition is absent. Example of false negative condition:: Person with disease was diagnosed with no disease.

There are several methods to evaluate machine learning models :

1. **Accuracy score**

   Classification accuracy is the number of correct predictions made as a ratio of all predictions made.

   The accuracy score is calculated by formula:

   $$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.14}$$

2. **Recall score**

   Recall score is the number of true positives divided by the number of true positives plus the number of false negatives.

   The recall score is calculated by formula:

   $$Recall = \frac{TP}{TP + FN} \tag{2.15}$$

3. **Precision score**

Precision evaluation method determines of how precise/accurate machine learning model of how many positives predictions have been predicted of total predictions. Precision score is calculated by formula:

$$Precision = \frac{TP}{TP + FP} \tag{2.16}$$

Precision is a good measure to determine, when the costs of False Positive is high.

4. **F1 score**

   F1 score is the harmonic mean of precision and recall taking both metrics into account. F1 score is calculated by formula:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.17}$$

5. **Confusion Matrix**

   A confusion matrix is a technique for summarizing the performance of a classification algorithm. Confusion matrix is a method to better understand the performance of classification models. It is a summary of of prediction results on a classification models: The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix.

3 Implementation

The implementation program main goal is to detect URL category of URL page HTML content. It is made by combining Natural Language Processing(NLP) and Machine Learning(ML) which are described in the Approach sections(links).

The implementation part consists of these sections:

1. The data set of URLs list
2. Data set creation using NLP techniques
3. Data preprocessing
4. Data training using ML techniques
5. Results and conclusions

The goal of the project is to solve the URL categorization task using a supervised classification algorithm and using a database with a large number of categorized websites. A supervised classified should be learned to predict the category a web page belongs to.

3.1 Data preprocessing

The one of the most important things solving Machine Learning kind of problems is valid data set source which is required for creating testing and training data sets which are required for training and testing Machine Learning model.

The data set of this problem is available at Data For Everyone data set lists. The data set contains 31086 di erent URLs with 81 attributes. Since the data set has a lot of attributes and few of them will be used in the implementation project, attributes could be represented in this kind of manner:

- *unit id*
- *golden*
- *unit state*
- *trusted judgments*
- *trusted judgments at*
- *main category*
- *main category:confidence*
- *sub category: Each category*
- *sub category: Each category:confidence*
- *url*

It is an enormous data set and all attributes is barely useful for the implementation problem, the size of data set attribute should be reduced. For the implementation part problem, 3 attributes must be extracted, other attributes should be excluded since they have no impact to the ML model creation.

List of attributes that are extracted for implementation part:

1. **main_category** : Attribute describes URL main category.
2. **main_category:confidence** : Attribute describes a probability of accuracy of the URL main category. Since URLs could have a lot of sub-categories it provides information about the most likely main category.
3. **url** : Attribute provides URL address of the web page.

The data set consist of 25 categories:

Adult	Finance	News & Media	Arts & Entertainment
Food & Drink	People & Society	Automotive	Gambling
Pets & Animals	Beauty & Fitness	Games	Reference
Books & Literature	Health	Science	Business & Industry
Home & Garden	Shopping	Career & Education	Internet & Telecom
Sports	Computer & Electronics	Law & Government	Travel

Table 2. The table of all possible categories

The primary data is not enough to determine in which category given URL belongs it needs some data preprocessing in order to make a vector and label set for Machine Learning algorithms.

Data preprocessing consist of 6 major steps:

1. Website scraping and text parsing
2. Text tokenization
3. Removing stop words and normalizing text
4. Website language determination
5. Words frequency
6. Chunk words

The whole image of the processes could be seen in the figure 6:

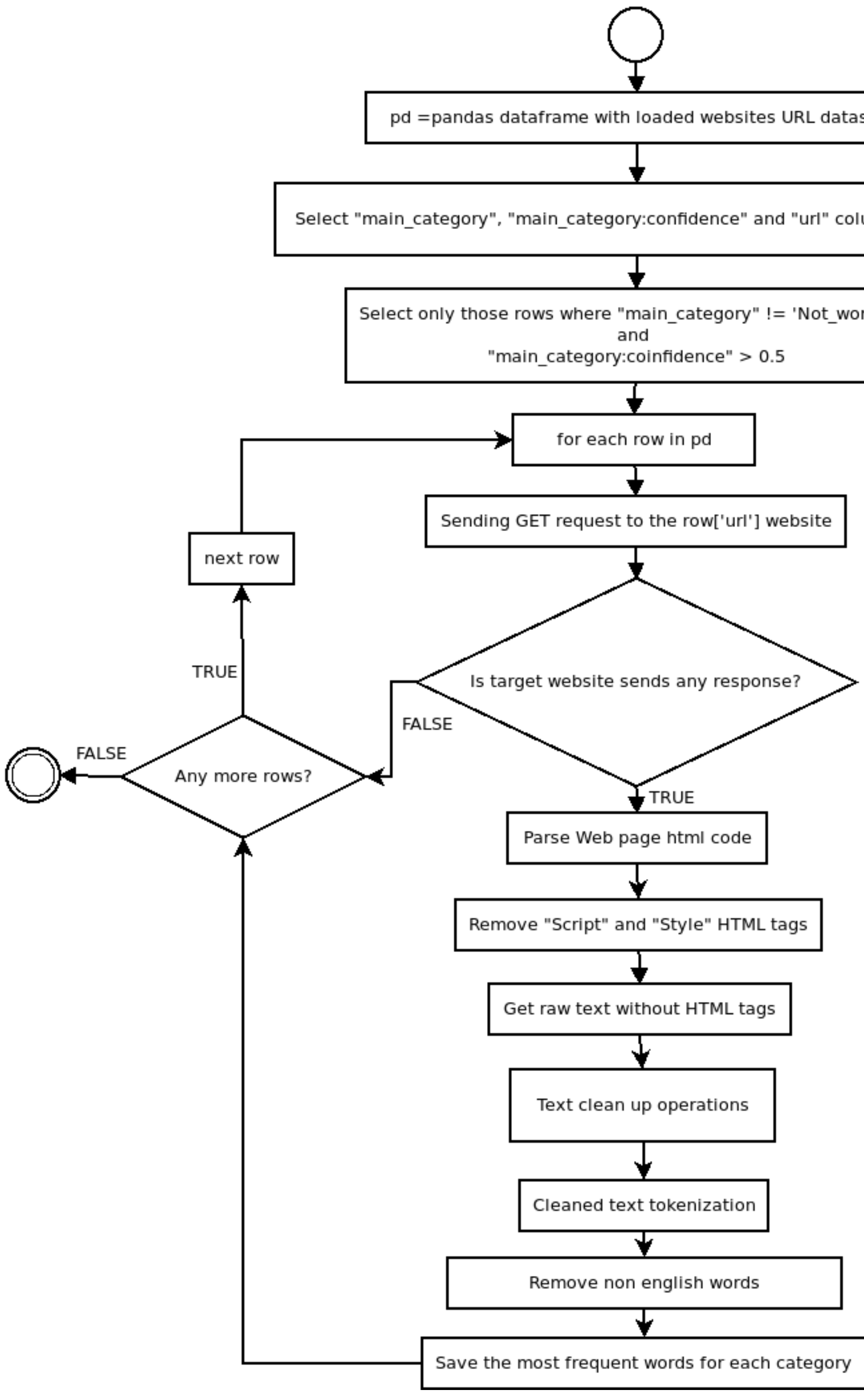

pd =pandas dataframe with loaded websites URL dataset
Select "main_category", "main_category:confidence" and "url" columns from pd
Select only those rows where "main_category" != 'Not_working' and "main_category:coinfidence" > 0.5
for each row in pd
Sending GET request to the row['url'] website
Is target website sends any response?
FALSE
TRUE
Parse Web page html code
Remove "Script" and "Style" HTML tags
Get raw text without HTML tags
Text clean up operations
Cleaned text tokenization
Remove non english words
Save the most frequent words for each category
Any more rows?
TRUE
next row
FALSE

Figure 6. Diagram of operations in the data preprocessing

Figure 6 represents steps that performs data downloading, data filtering and data clean up operations. Since there are a lot of process in this algorithm, in this implementation part each process would be described separately in more depth.

3.1.1 Website scraping and text parsing

The first process is to analyze website and download its content. Every website is build up using HTML language as a core code and the additional design and performance features are added using CSS and JavaScript tools.

The category of the website determines website content. The content is written in the HTML programming language using special tags for displaying information for a user. The main goal of this process is to download the content and parse only necessary text which would be useful for this project.

This section consist of two parts:

1. **Downloading website content**

   Each website content is downloaded by sending GET request using python library Urlib. This library allows to perform GET request to the website and tracks it response.

   The expected response from the website is HTML code including Cascading Style Sheets(CSS) and JavaScript(JS) codes. However some websites have security features from scrapping programs, so it is not always possible to get content information from the website. Each website could see the program request header and determine that the request is not performed by the ordinary user and in this case some websites do not respond with the proper information.

   Before sending an GET request to the website, the request headers properties are changed to the more likely normal browser properties in order to prevent websites for returning inappropriate response with the HTML content. Also the timeout of the request headers is set to the 15 seconds in order to make some additional time for website response.

   The website is skipped if it is not responding to the GET request

2. **Website content text parsing**

   Website response contains HTML code where the necessary text is stored. There are no need to store text within HTML tags so the other step is to leave only text by excluding HTML tags. This is made by using python library beautifulsoup. Text parsing consist of these steps:

   (a) Parse text into beautiful soup object for text filtering and manipulation properties

   (b) Filter out CSS and JavaScript content. CSS language is additional tool for improving HTML design which is marked with <*style*> HTML tag and JavaScript is programming language for website behaviour which is marked with <*script*> HTML tag. These elements usually do not contain any useful information so these HTML tags are removed by using beutiful soup function *decompose()*

   (c) Text of remain HTML code is extracted by removing HTML tags. HTML tags are removed by using beautiful soup function *get_text()*

   (d) New lines symbols are removed from the text

   (e) The extracted text is stripped in order to remove empty symbols in the text

   (f) Every word is joined to the text string by a new line

3.1.2 Text tokenization

The plain text of the website content is not proper format for this project because there is a small probability that the text would be the same among di erent websites with the same category. However the probability is much bigger that the separated words would more likely to be similar among other website with the same category. This is the reason why the text should be split up to the tokens which would perform plays an important role into project development.

The parsed text of the website is split up into tokens by using regular expression word tokenizer. Regular expression tokenizer is expressed by this regular expression query:

`((?<=[^\w\s])\w(?=[^\w\s])|(\W))+$.`

Regular expression symbols explanation:

`?` A question mark matches zero or one of the preceding character, class, or sub-pattern.

`<?=` Is a positive lookahead, a type of zero-width assertion. What it's saying is that the captured match must be followed by whatever is within the parentheses but that part isn't captured.

\ Classes of characters: The square brackets enclose a list or range of characters (or both).

^ May appear at the beginning of a pattern to require the match to occur at the very beginning of a line.

`\W` Matches any single "word" character, namely alphanumeric or underscore.

`\s` Matches any single white space character, mainly space, tab, and newline (‘r and ‘n). Conversely, capital §means "any non-white space character".

+ A plus sign matches one or more of the preceding character, class, or sub-pattern.

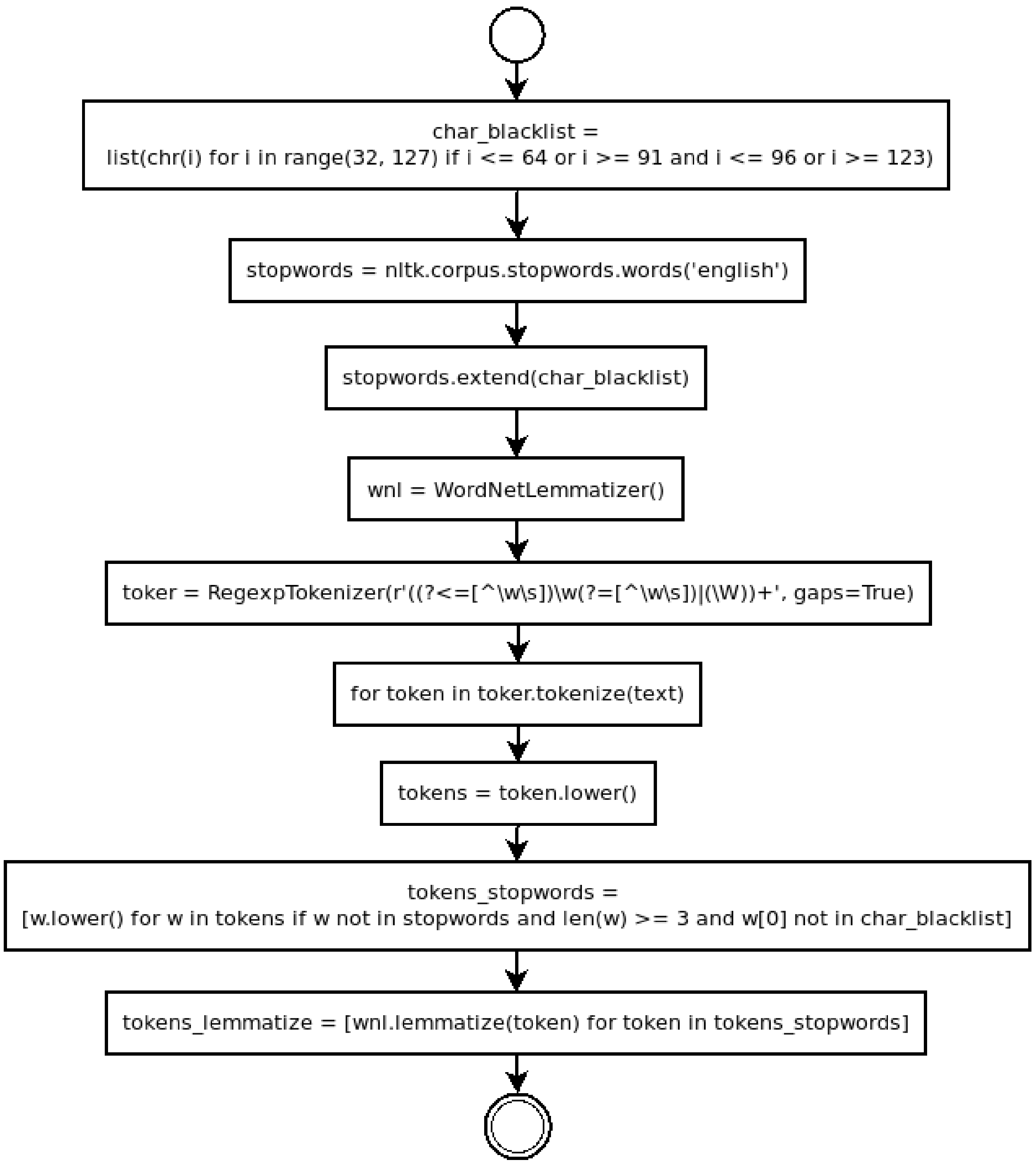
This regular expression is passed to the nltk python library RegexpTokenizer function giving an plain text as parameter and it converts plain website text into tokens.

When the text is tokenized, each token word is converted to the lower case. This is made because upper and lower cases have di erent ASCII codes, so the computer the same words with di erent ASCII codes treats words as di erent despite that the words have the same meaning.

3.1.3 Removing stop words and normalizing text

Tokens of the website text contains a lot of words, but this does not mean that all the words are proper to use for creating a Machine Learning model. At this point tokens could consist of stop words and random symbols and word could be written in a di erent grammar forms(singular and plural). These words take a lot of volume in the data, so the volume should be reduced by performing text normalization operations such as **Removing stop words** and **Words lemmatization**.

The full image of this process could be seen in a implementation code flowchart figure(7):



```
char_blacklist =
list(chr(i) for i in range(32, 127) if i <= 64 or i >= 91 and i <= 96 or i >= 123)
stopwords = nltk.corpus.stopwords.words('english')
stopwords.extend(char_blacklist)
wnl = WordNetLemmatizer()
toker = RegexpTokenizer(r'((?<=[^\w\s])\w(?=[^\w\s])|(\W))+', gaps=True)
for token in toker.tokenize(text)
tokens = token.lower()
tokens_stopwords =
[w.lower() for w in tokens if w not in stopwords and len(w) >= 3 and w[0] not in char_blacklist]
tokens_lemmatize = [wnl.lemmatize(token) for token in tokens_stopwords]
```

Figure 7. An algorithm for processing stop words exclusion and words lemmatization

1. **Removing stop words**

   Stop words are considered to be common words in a human language which do not provide lots of information by using itself. Tokens contains a list of words that means that they are not connected to the each other and some words by itself do not provides useful information. This is the reason why these words should be removed from the token list because they could confuse the model since most likely every category would contain the same set of words (stop words).

   English language is considered of having around 180 stop words. The stop words list database could be found on the internet but for this project stopwords.words('english') function which contains 179 commonly used english language stop words.

The main goal of this process is to make sure that the words in the tokens would be meaningful as much as possible, so the additional rules of stop words determination is added. The words with length less than 3 is also considered as as stop words because in english language there are not many words that would contains meaningful information. This is done in order to prevent random symbols that is not determined as words in the english language. The list of these kind of random symbols is made by using ASCII(American Standard Code for Information Interchange) table. The symbols which are in the range of the ASCII table of: [32 : 64], [91 : 96], [123:127] are considered as the stop words. The integer numbers from the range are generated and they are converted to the char symbols that are saved into the list. The list of these symbols are:

’ , !, ", , $, %, &, ’, ,, , , ,, -, ., /, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, :, ;, <, =, >, ?, @, [,], ^, _, ‘, , |, , ’

When the stop words list is generated then it is possible to filter out stop words from the tokens. The pseudo algorithm of stop words elimination from the token list:

(a) Each word from the token list is iterated

(b) If word is in stop words list then this word is excluded from the token list else the word remains in the token list

2. **Words lemmatization**

Words lemmatization in linguistics is the process of grouping together the inflected forms of a word so they can be analyse as a single item, identified by the word’s lemma, or dictionary form. English words grammatically could have 2 forms - singular and plural form. The words core meaning of singular and plural forms remains the same, but the program would treat them as totally di erent words. The words form is not important for this project problem, so the goal of this process is to have one form for all words that these words would be treated equally according to their core meaning.

The words lemmatization is done by using english words dictionary of singular and plural forms. The dictionary is build in in the nltk python library WordNetLemmatizer function. The function takes a word as argument and returns a singular form of the word, for example:

$children \longrightarrow child$

$word \longrightarrow words$

$word \longrightarrow word$

The examples demonstrates how words forms are converted. The rules of word lemmatization:

- If input word is in a **singular** form, then output would be an input word in a **singular** form
- If input word is in a **plural** form, then output would be an input word in **singular** form

3.1.4 Website language determination

There are many human languages with the di erent words, grammatical features and letters. Each language requires their own approach and analysis, so for this project the one main language is chosen - English language. This language have been chosen, because the majority of websites in the data set contains english language content.

English language considered to be dominant language among websites, because **10436/19922 (52.38 %)** are english content websites which are still functioning and the category of these websites is known.

Language detection could be trivial, because some websites could contain more than one language in their content. For this project the english language was chosen and the language determination was proceed by calculating the ratio of total website words number and website english words number. If the ratio value is more then 50 %, then the website is considered to be english.

Since some websites could contain more than one language, for this project the english words dictionary is used which is produced by nltk library english dictionary data set. The dictionary contains around 171,476 english words which are popular and used every day in colloquial language, 47,156 obsolete english words and 9,500 derivative english words. English words dictionary and website words tokens is enough to calculate ratio of english words and exclude non english words from website tokens list. The ratio of english words in the website is not enough, because it could appear websites, where all words are in english, but with small number of instances. These kind of websites are not really useful, because they are lack of words so the websites with less then 10 english words are excluded from the websites list.

Implementation of the language detection could be seen in the code flowchart figure(8):

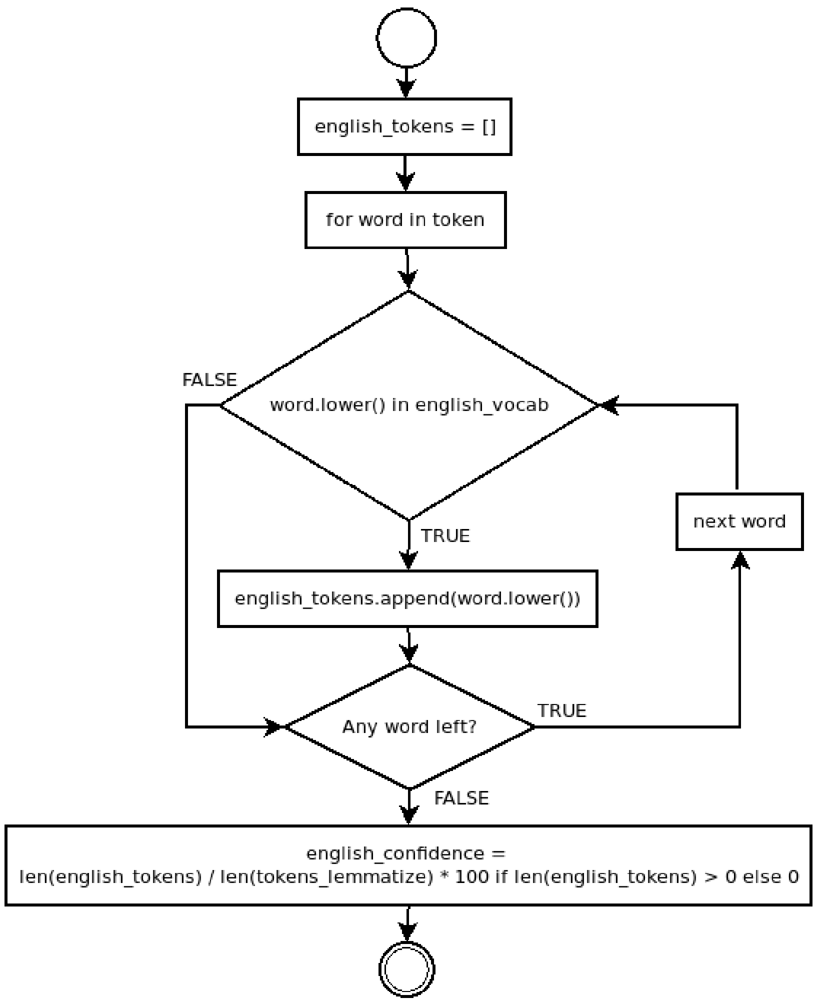

english_tokens = []
for word in token
word.lower() in english_vocab
FALSE
TRUE
next word
english_tokens.append(word.lower())
Any word left?
TRUE
FALSE
english_confidence =
len(english_tokens) / len(tokens_lemmatize) * 100 if len(english_tokens) > 0 else 0

Figure 8. An algorithm for creating words list

Pseudo algorithm for website language determination process:

1. A list which would contain all english words is initiated
2. Each word from website tokens list is iterated
3. If word appears in the english words dictionary, then word is appended to the english words list(step 1)
4. Steps 1, 2, 3 are repeated until all words from the token list are iterated
5. The ratio of english words in the website is calculated by dividing number of english words to total number words in the website token and multiplying by 100 to get percentage representation
6. If the ratio is less then 50 % and webpage contains less than 10 english words, then the website considered to be non english content and it would be excluded from the website list

English words which was appended into english words list in step 1 is considered as new words token representation for website since all non english words are excluded.

English websites consist of 101 di erent top-level domains(TLD). The diversity of the 10 most frequent domains could be seen from the figure(9) :

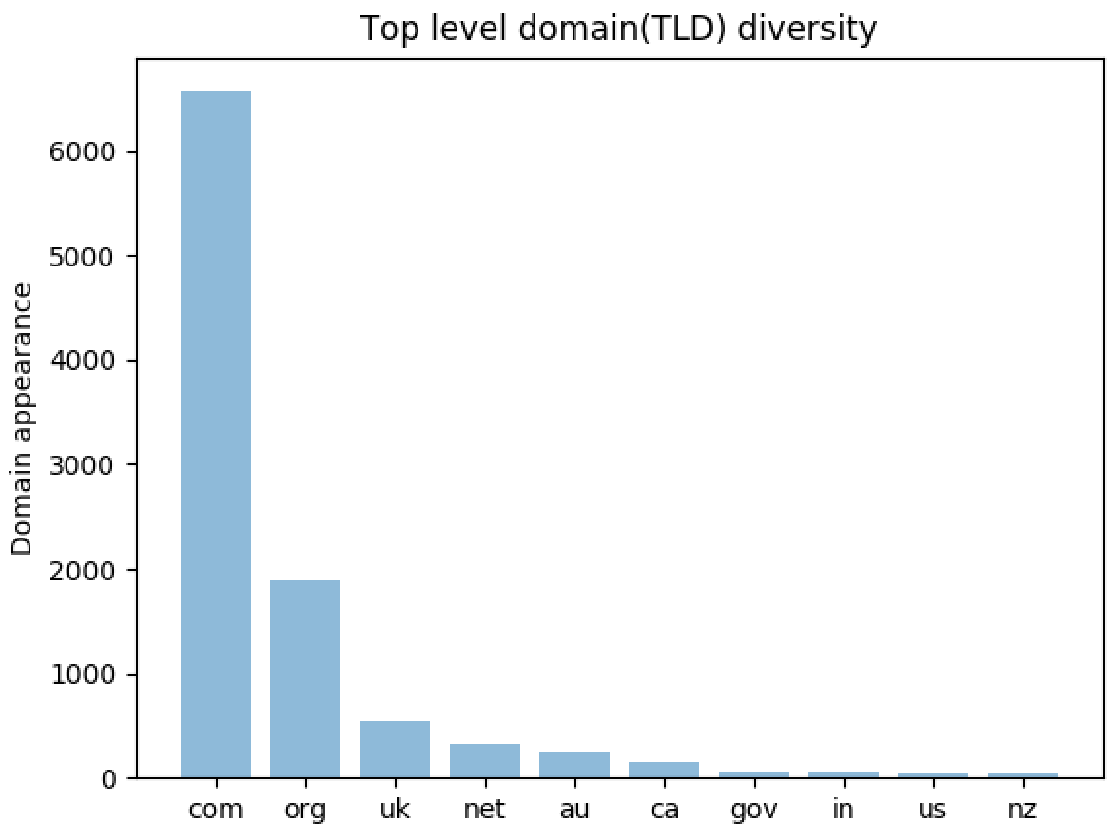


Figure 9. Top Level Domain(TLD) diversity in the english content websites

According to the generated results of TLD diversity in the english content websites (9) the most frequent domain is **com** with *6571* instances, the second place took **org** domain with *1885* instances and the third place took **uk** domain with *540* instances.

3.1.5 Words list creation

In theory every category should contain a di erent set of words that could represent category itself. At this point a new generated english words token list from the previous subsection contains words for particular website, but not for particular category. One website data is not enough to define the list of the most frequent words in a category, but all websites data are able to create a total word list of all websites for each category where the most frequent words list will be generated after all websites would be analyzed.

The total words list for 24 categories is generated by appending each website words from token list. The dictionary data type is created with shape of *(24, x)* where first dimension represents the name of category and the second dimension represents the set of total words for each category. The value of second dimension is equal x, because each category could contain di erent number of words.

Implementation of the total words list creation could be seen in the code flowchart figure(10):

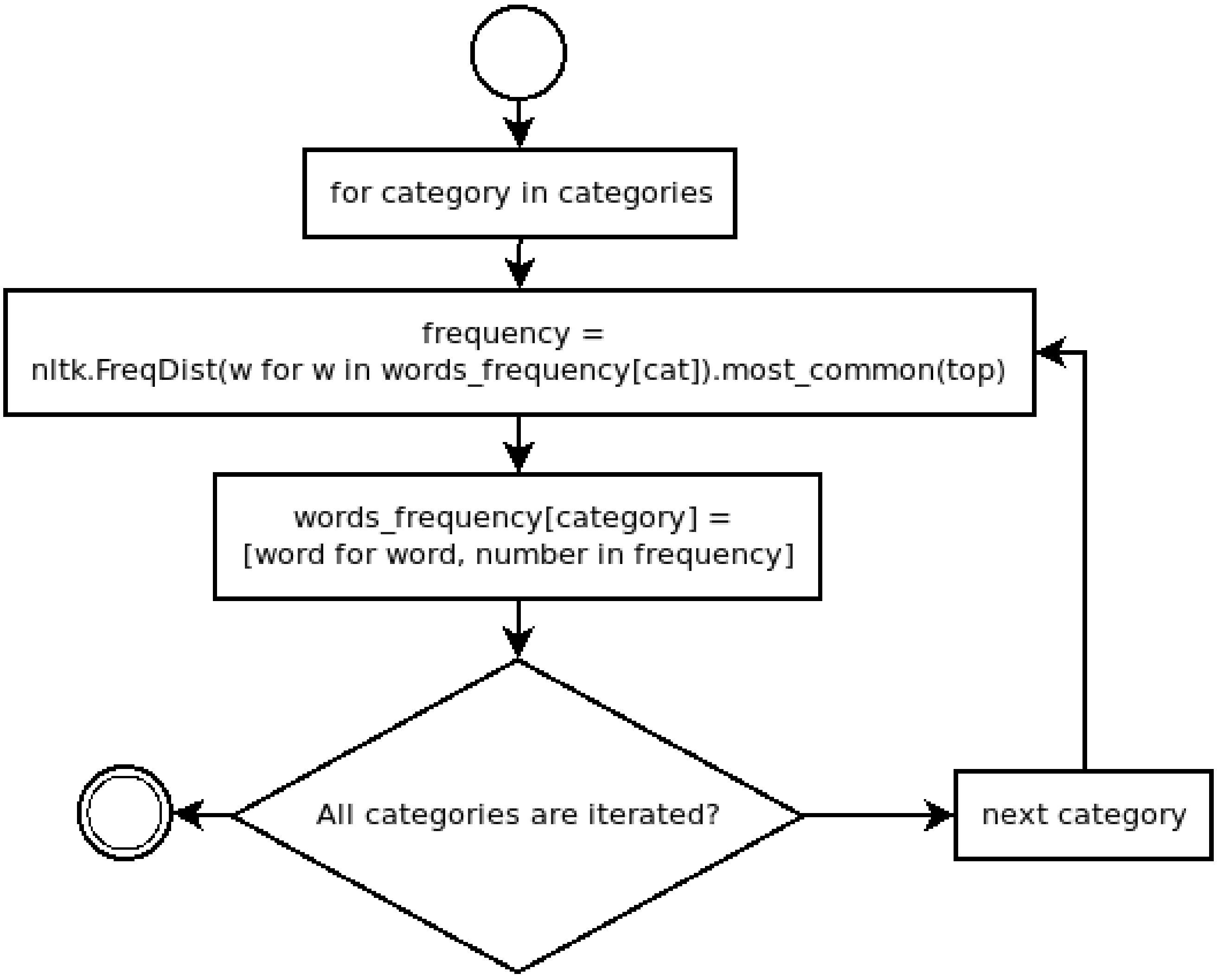

for category in categories
frequency = nltk.FreqDist(w for w in words_frequency[cat]).most_common(top)
words_frequency[category] = [word for word, number in frequency]
All categories are iterated?
next category

Figure 10. An algorithm for creating words list

Total words list generation pseudo algorithm:

1. Each category is iterated from the categories list
2. If the dictionary do not have key instance of the current category, then dictionary with category key is initialized and it equals for empty list
3. The dictionary list of category key is extended with a website words token list
4. Step 1, 2, 3 is repeated until all categories are iterated

Total words list for each category will take a place for creating a list of the most frequent words list for each category.

3.2 Words frequency model

This part would cover the most frequent word generation for each category. At this point, the words list should be generated from all websites words tokens including their categories. The words list contains all existing words for each category, so the main part of this section is to determine the most frequent words and sort them ascending mode. This list of the most frequent words would be used for the custom model prediction and also for features generation for Machine learning model.

This section consist of 2 parts:

1. **The most frequently words list for categories** - a list of the most popular words list for each category
2. **Chunk words** - methods for detecting chunk words and exclude them from the most frequent words list

3.2.1 The most frequently words list for categories

All websites are classified into 24 categories. Each category has it is own features which are defined by the words, for example: set of words ['World', 'Business', 'Science', 'Weather', 'Culture', ...] implies that it is probably a set of keywords that may belong to "News Media" category. Humans could categorize websites content could classify intuitively if website keywords match well known patterns of the category.

The most frequent words list is made from the previously generated word list - all possible words for each category is stored into this list so the main task for this phase is to calculate the frequency of the words and sort them in ascending mode. The vector of the most frequent words list would consist of 2500 words for each category.

The shape of the most frequent words list: *(24, 2500)* in other words: *categories list, 2500 the most frequent words).* It is possible to have lower shape of the most frequent words if category all websites contains less then 2500 di erent words in that case number would be equal a set of words available in a category websites contents.

Implementation of the most frequently words list for categories creation could be seen in the code flowchart figure(11):

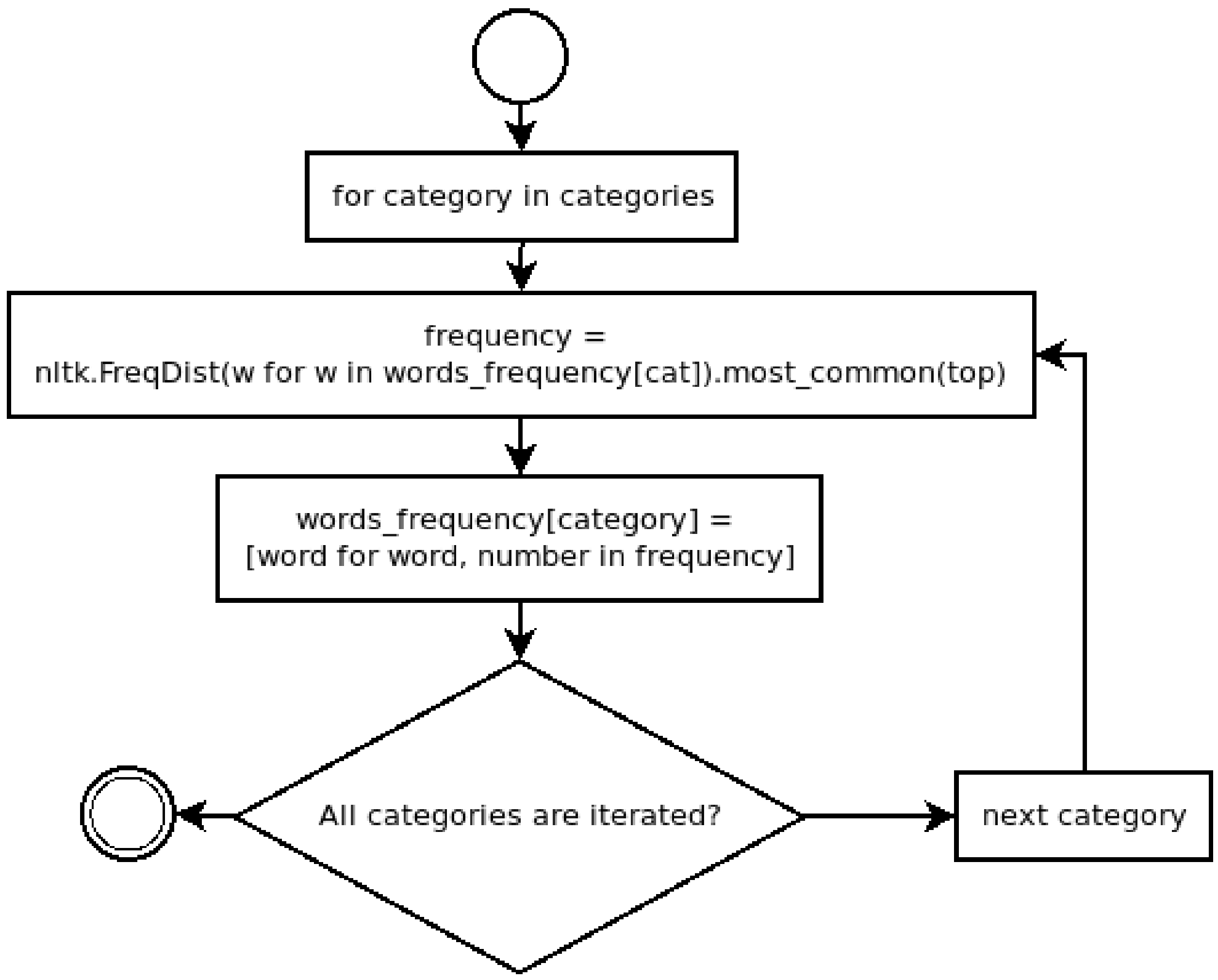

for category in categories
frequency =
nltk.FreqDist(w for w in words_frequency[cat]).most_common(top)
words_frequency[category] =
[word for word, number in frequency]
All categories are iterated?
next category

Figure 11. An algorithm for creating the most frequent words list for each category

The pseudo algorithm how the most frequent words list is made:

1. Each category is iterated from the categories list
2. Calculating words frequency for each category. The frequency is calculated by using nltk.FreqDist function. A function requires a string data type as parameter and it returns the tuple data type of words and their frequency occurrence. The input of the function is words list for each category. Each word from the words list is iterated and passed to the function. In this way the function returns word with it frequency. The words and frequency is sorted in ascending mode. Example of an output of TOP 5 most frequent words in 'Books_and_Literature category': $[('book', 5625), ('text', 4094), ('new', 2322), ('novel', 2009)...]$.
3. Extracting words from the frequent words list. Since the only goal of this process is to generate a list of the most frequent words in the category, the number of words frequency would not be used for the Machine Learning model so only words are extracted and saved in the words frequency list.

3.2.2 Chunk words

Generated words frequency list is not perfect because websites could contain a lot of chunk words which are common to all categories. Content of the websites could depend on many factors -

current month, political news, casual news or default HTML home page design which includes words like contact or home. Words that are in the TOP 15 the most frequent word list and appears more or equal than 7 categories are called chunk words.

Implementation of creating and removing chunk words could be seen in the code flowchart figure(12):

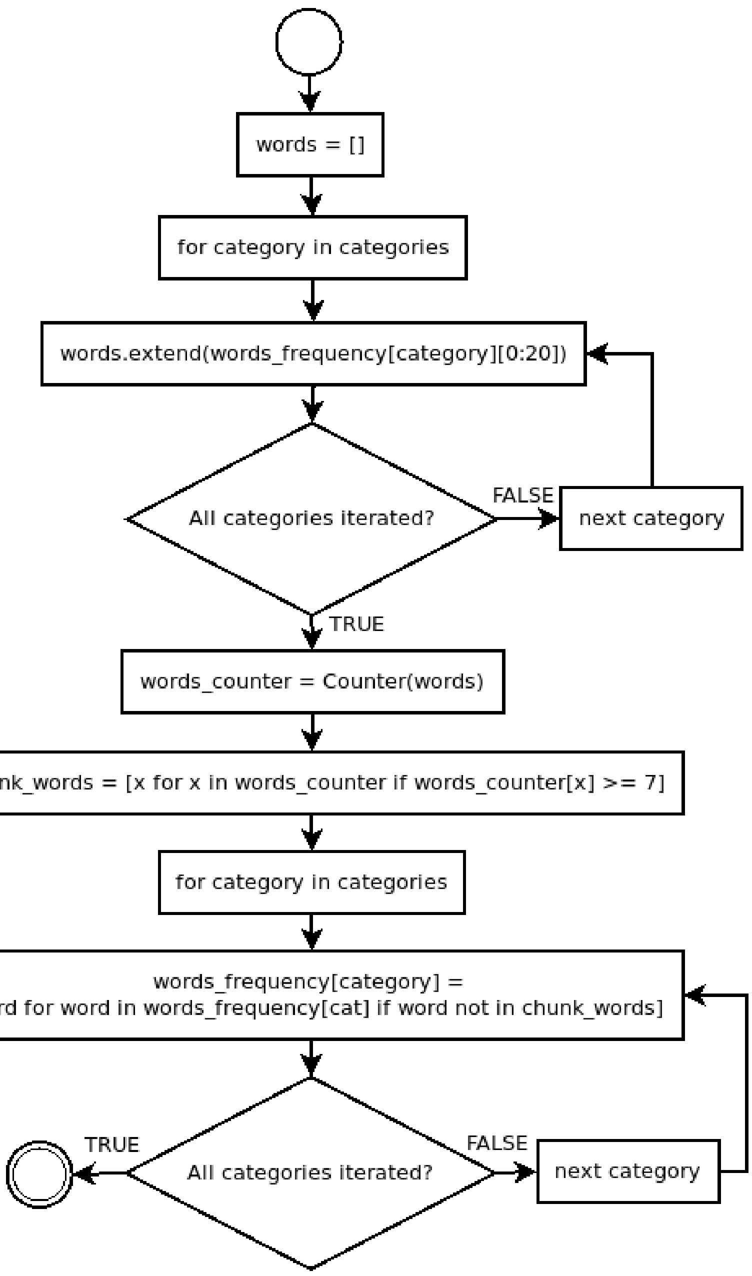

words = []
for category in categories
words.extend(words_frequency[category][0:20])
All categories iterated?
FALSE
next category
TRUE
words_counter = Counter(words)
chunk_words = [x for x in words_counter if words_counter[x] >= 7]
for category in categories
words_frequency[category] =
[word for word in words_frequency[cat] if word not in chunk_words]
TRUE
All categories iterated?
FALSE
next category

Figure 12. An algorithm for detecting and removing chunk words

The pseudo algorithm for detection of chunk words and chunk words removing process:

1. An empty list *words* is created where all top 15 words would be stored
2. Each category is iterated from the categories list
3. Top 15 most frequent words in each category is extended to the words list (Step 1)
4. Steps 2 and 3 are repeated until all categories would be iterated

5. Words are counted by using Counter function
6. If word appears more or equal than 7 times in di erent categories, then word is added to the chunk words list
7. Each category is iterated from the categories list
8. Words are excluded from the most frequent words list for category by checking if word is in the chunk words list

All chunk words could be seen in the figure(13) :

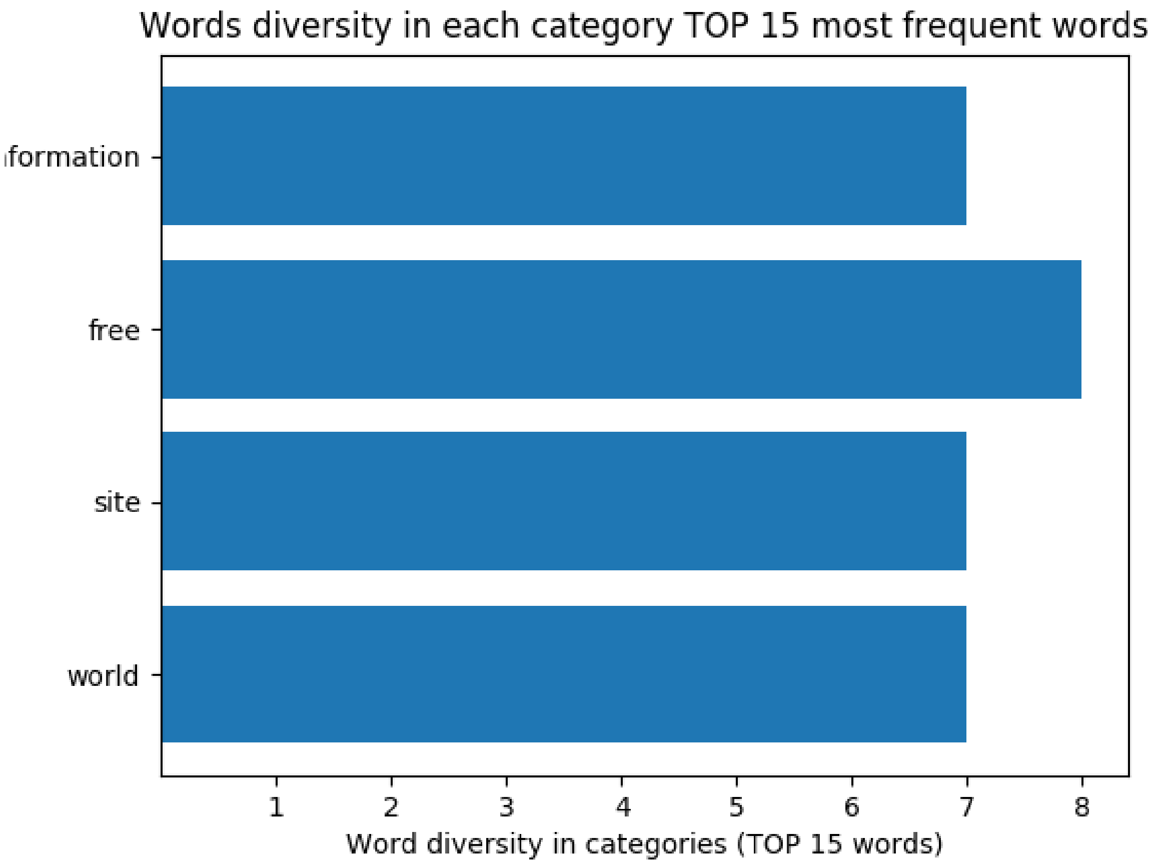


Figure 13. An diagram of chunk words diversity in categories

According from the most frequent words from each category data generated in December, the bar chart figure(13) shows that there are 4 chunks words: *information, free, site, world*. The word ’free’ appeared in a 8 categories while other words appeared in 7 categories. These words are excluded from the most frequent words list.

3.3 Custom model

Machine Learning approach is not only way to predict category of the website - other way is to use raw statistics and create a prediction model from already generated words frequency list of each category which was created in the previous section. Since it is known categories most popular words it is possible to find a pattern and predict a category of URL without Machine Learning approach. Custom model creation requires 2 steps:

1. Categories features generation
2. Calculate words weight

This method do not require machine learning approach and it is capable of predicting category of the target website (a website which category will be predicted). Method performance results are much better than usual Machine Learning model. The results and conclusions of custom model are described in the "Results and Conclusion settings" section.

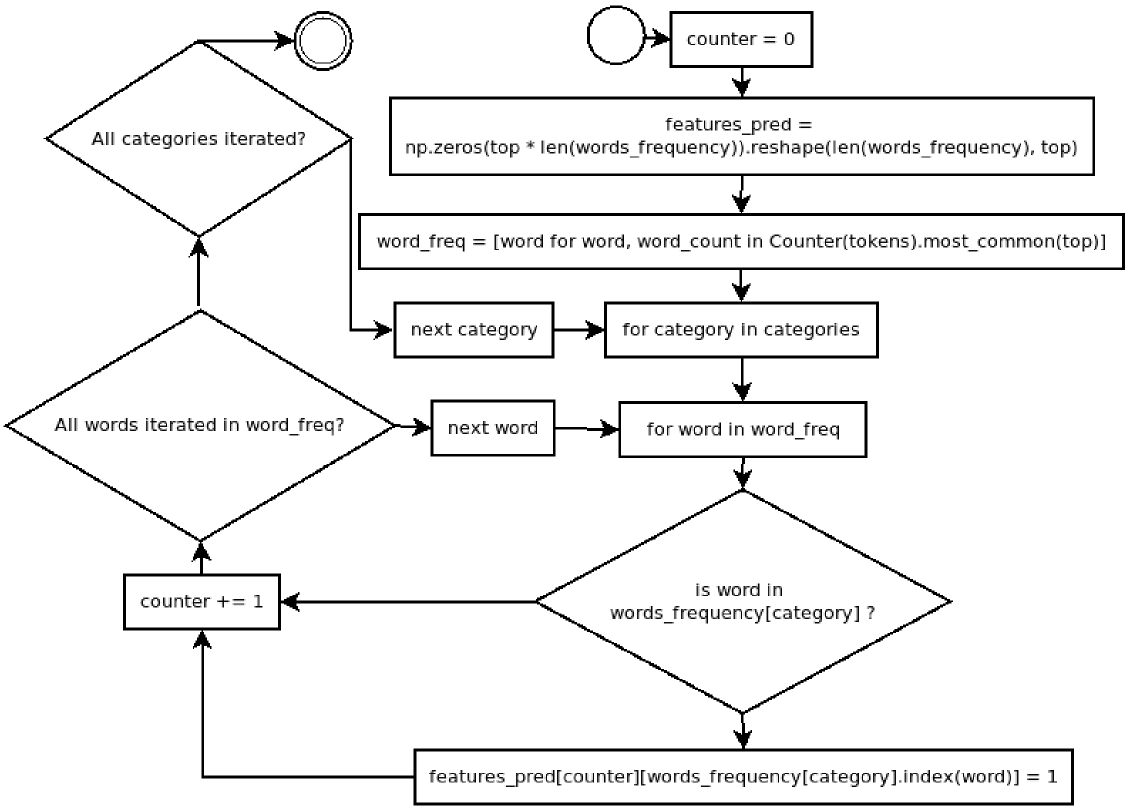
3.3.1 Categories features generation

The most frequent words list for categories is used not only for Machine Learning, but also for categories features generation. The method requires a *target website most frequent word list* and *categories most frequent words list* in order to generate the categories features list to determine a target website category.

The first step of the model creation is gathering the most frequent word list of a target website. The main goal of this process is to create a most frequent words model for each category. Each category would have a list of words that occurred in the target website. At the beginning, each word in the list is equal to the value of 0. By going through each category, if the word occurs in the most frequent words list, then that position the value changes from 0 to 1.

The idea behind this process is to generate vector with the shape: **(categories number, number of top words in category)**. In implementation part there are 24 di erent categories and there are tracked top 2500 the most frequent words, so the shape of the list should be: **(24, 2500)** if there are no chunk words excluded, otherwise it would depend on the particular category most frequent words list length.

Implementation of this process could be seen in the code flowchart figure(14):



counter = 0
features_pred = np.zeros(top * len(words_frequency)).reshape(len(words_frequency), top)
word_freq = [word for word, word_count in Counter(tokens).most_common(top)]
for category in categories
next category
All categories iterated?
for word in word_freq
next word
All words iterated in word_freq?
is word in words_frequency[category] ?
counter += 1
features_pred[counter][words_frequency[category].index(word)] = 1

Figure 14. Algorithm of creating the most frequent word vector

The Categories features generation for the website pseudo algorithm:

1. Creating integer variable "counter" which would track a category index of the list. At the beginning "counter" variable value is equal to 0
2. Creating a list of the most frequent words: "frequent_pred". Later on this variable would store of all possible combinations of each category of the most frequent words list.

   For each category values of the list are set to 0.

   Variable shape: (25, 2500). First dimension of the list marks category, second dimension of the list marks the top 2500 most frequent words.
3. Storing top 2500 the most frequent words in the "words_freq" variable. The website all words are counter and sorted in the ascending order from the most frequent word to the least frequent word of top 2500 words.
4. Iterating for each category from the categories list
5. Iterating for each word from the "words_freq" variable
6. Checking if the current category most frequent words list contains website current word. If it contains, the index of the word position in the words_frequency list for current category is saved and used by changing a value of features_pred[counter][index] from 0 to 1
7. Step 4 repeats until there no unchecked words left in the word_freq list
8. Counter value is auto incremented by 1. This action means that the category is changed to the next one
9. All steps above are repeated until all categories from the categories list are iterated

In the result of this process the list of the most frequent website words for each category is created. At the beginning the vector is filled with 0 values, but if the word of the website occurs in the category top frequent words list, then the value at the particular position is changed to 1.

In the end of this process, the list of frequent words features is generated for each category. The website words have an impact of a list values. This process generates a list of possible features for each category so the next step is for deciding which feature suits model the best and it is first dimension value would be considered of a target website category.

3.3.2 Categories weight estimation

Model feature list which was generated in the previous section should be evaluated in order to extract the category of the target website. The feature estimation is done by processing weight calculation of each category of model feature list and then choosing the max value of the calculated categories which would be considered as category of the target website.

The weight estimation of the model feature list is done by iterating each category list and checking the position whenever values are equal to 1. Every position should have their own weight, because the word frequency list is sorted in ascending mode so words that are close to the list beginning should be more important than those words that are near list end. The main formula of calculating weights of each category is:

$$weight\_sum = top - index \tag{3.1}$$

weight_sum A total sum of all words weights in the category

top is a value of how much words are in the most frequent words list which by default value is equal to 2500

index An index of where the value of 1 occurred in the category features list

The process of category weight estimation flowchart could be seen in the figure(15):

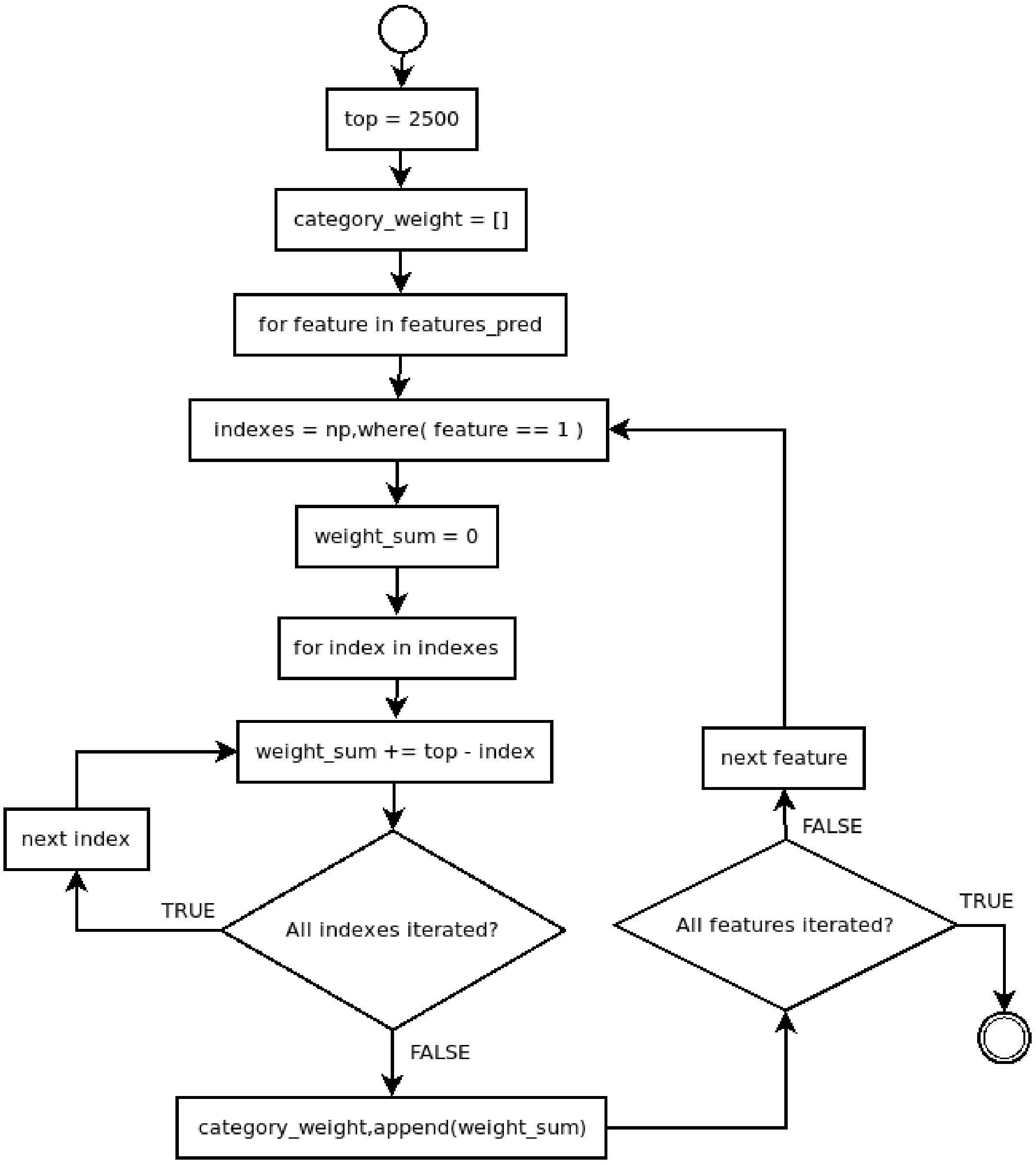


Figure 15. An algorithm of categories features weight estimation process

Category weight estimation pseudo algorithm:

1. Creating a list where each category weight sum would be stored. The list is named as *category_weight*
2. Iterating for each category features from the categories features list
3. Getting all positions of the category features list where value is equal to 1. This is done by using unction. This function requires an input of condition and it generated the position

indexes as an output. The condition as an argument to the unction: $category\_feature == 1$. The function iterates each value of the category features list and if that value is equal to 1, then it stores that value index in the category features list. The output of the function is stored to the *indexes* variable as a numpy data list type.

4. Creating an integer type variable for storing weight sums for category features list. The variable is named as *weight_sum*. At the start this variable value is equal to 0
5. Iterating for each index of *indexes* numpy list
6. Calculating a weight sum for category features: $weight\_sum+ = top - index$
7. Step 6 is repeated until all indexes are iterated
8. The category features weight is appended into *category_weigh* list
9. Steps 2, 3, 4 ,5, 6, 7, 8 are repeated until all categories features are iterated

The main point of this process is to generate a list of all weights for each category features. That list is named as *category_weight*. At this point the main task is to track the category from the categories features list of the highest weight sum by using unction which returns the highest value in the list. As an input function requires a list and as an output it returns the highest value in the input list. The output value of the highest category features weight sum is indicating website category by using unction which returns the index of the input value: $category\_index = category\_weight.index(max(category\_weight))$

3.4 Machine Learning

In this section Machine learning algorithms would be trained and tested using by already created the most frequent words list for categories, websites words data and websites categories. The main point of this section is to create a fully working Machine Learning model which would be able to predict a website category without no human interaction for the primary data set.

This is done by processing 4 steps:

1. Features and labels creation
2. Machine Learning models training
3. Models performance evaluation

3.4.1 Features and labels creation

Machine learning algorithms models requires a proper form of data which is the main key of the successful performance of Machine Learning models. At this point the data is almost set up properly - the most frequent words list for categories is created which would help in creating a features and labels for Machine Learning model. Features and labels definitions:

Features are attributes or properties shared by all of the independent units on which analysis or prediction is to be done.

In this project the features would be considered to be binarized websites most frequent words lists according to the most frequent words list for category list. If a target website most frequent words list values are in the most frequent words list for specific category list, then the value in the most frequent website words list is changed 1 in a position of a word, otherwise value is set to 0.

Data set contains the categories of all websites, so in this case it is known which category of the most frequent words list for categories has to be used.

Labels are the final output.

In this project labels are the categories of websites. Since the data set contains all categories of websites so the main task is to track them properly and match them with the features that Machine Learning model would know category of particular website feature set.

The process of features and labels creation for Machine Learning model flowchart could be seen in the figure(16):

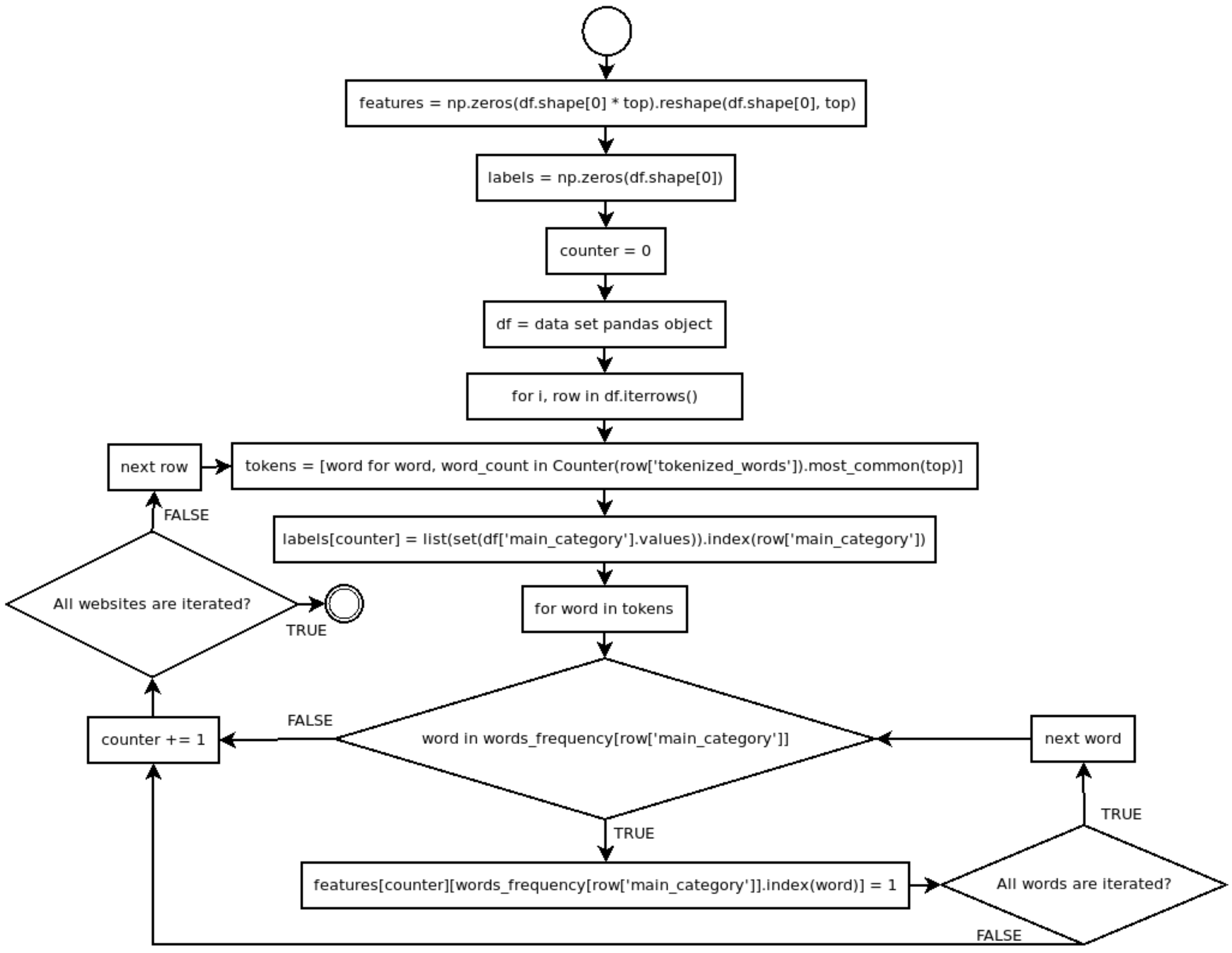

features = np.zeros(df.shape[0] * top).reshape(df.shape[0], top)
labels = np.zeros(df.shape[0])
counter = 0
df = data set pandas object
for i, row in df.iterrows()
next row
tokens = [word for word, word_count in Counter(row['tokenized_words']).most_common(top)]
FALSE
labels[counter] = list(set(df['main_category'].values)).index(row['main_category'])
All websites are iterated?
TRUE
for word in tokens
FALSE
counter += 1
word in words_frequency[row['main_category']]
next word
TRUE
TRUE
features[counter][words_frequency[row['main_category']].index(word)] = 1
All words are iterated?
FALSE

Figure 16. An algorithm of features and labels creation

Features and labels creation pseudo algorithm:

1. List named *features* is created for storing websites features. The features list has a shape of **(english website length in the data set, top)** which is equals to **(10436, 2500)** by using december websites data set.

2. List named *labels* is created for storing websites labels. The labels list has 1-dimensional shape of **(english website length in the data set)** which is equals to **(10436)** by using december websites data set.

3. Integer variable *counter* is created for tracking an index of *labels* list and 1-dimension of *features* list. At the beginning this variable is set to be equal 0

4. Iterating for each website(*row*) and their position(*i*) in the december websites data set

5. Defining *labels* list value in the position of *counter*. The category of the website is saved into *labels* list of *counter* position

6. Iterating for *word* in website token list

7. Checking if *word* is in the most frequent words list for the current category - if it is true, then *features* list value in 1-dimensional position of *counter* and 2-dimensional position of word in the most frequent words list for current category is set to 1

8. Steps 7 is repeated until all words are iterated from a website token list

9. *counter* value is auto incremented by 1

10. Steps 5, 6, 7, 8, 9 are repeated until all websites are iterated from the december websites data set

Features and labels are the main data that would be fit to Machine Learning models. However it is a bad practise to use all this data on training machine learning models because of these factors:

1. Testing machine learning model. If all data would be used for machine learning model training, then how to be sure how well the model is performing in predicting data set if there are no testing data? Testing data is an indicator of how well machine learning model is performing.

2. Over fitting problem. Over fit is when your training model is too precise and doesn't generalize on the problem you try to solve. In other words it is too fit for the training, and the training alone, so that it cannot solve/predict a di erent data set.

These factors are the reason why features and labels data should be split into **training** data and **testing** data.

At first features and labels data are shu ed by using sklearn.utils.shu e(*arrays, **options) function. It shu es arrays or sparse matrices in a consistent way. Function main argument is features and labels data sets which should be shu ed.

The next step of the shu ed data create training and testing data sets by using sklearn.model$_{s}election.tr$
$options)$ function. This function splits arrays or matrices into random train and test subsets by proportion of 67 % of total data to training data, and 33 % of remaining data to the testing data.

3.4.2 Machine learning models creation

Building machine learning models from the scratch requires a lot of time and advanced knowledge about concepts of a particular machine learning model parameters implementations. Machine learning models for implementation part would be created by using Machine learning libraries where a lot of models are already build in. Models are created by using scikit-learn library for python programming language. scikit-learn python library has build in already prepared ready to use machine learning models.

In the implementation part there are created 2 di erent approach supervised machine learning classifiers types:

1. **Logistic Regression (LR)**

   Logistic regression model is created by using ibrary. Fundamentals of the Logistic Regression model are described in the Logistic regression section(1a)

2. **Linear Support Vector Classification (LSVC)**

   Linear Support Vector Classification is one of the branch of Support Vector Machine (SVM) classification model in other words - they are di erent implementation of the same algorithm.

   The di erence between LSVC and SVM: A regular SVM with default values uses a radial basis function as the SVM kernel while LSVC uses a linear kernel for the basis function. That makes LSVC model much less tunable and is basically just a linear interpolation.

   Linear Support Vector Classification is created by using ibrary with default parameters. LSVC parameters are almost identical as Logistic regression model, except:

Each model has their properties that would have an impact to the model performance. In Logistic Regression and Linear Support Vector Classification shares almost identical parameters. Some of the models parameters description:

- *C = 1.0.* C parameter is inverse of regularization strength. Smaller values of C specify stronger regularization. This variable value must be positive. C value is set to 1.0
- *class_weight = None.* Weights associated with classes in the form {class_label: weight}. This value is set up to 'None', because all classes supposed to have at least one weight
- *penalty = 'l2'.* Used to specify the norm used in the penalization. L2 penalty indicates that model uses *Ridge Regression* regularization technique.
- *dual.* Dual or primal formulation. Dual formulation is only implemented for l2 penalty with liblinear solver. Prefer dual = False when n_samples > n_features. In **Logistic Regression** model dual parameter is set to False, while in **Linear Support Vector Classification** model dual parameter is set to True.
- Fit_intercept = True. The intercept is the expected mean value of features when all labels are equal to constant value.
- intercept_scaling = 1. Intercept scaling is used to lessen the e ect of regularization on synthetic feature weight.

- max_iter = 100. Maximum number of iterations taken for the solvers to converge.
- random_state = None. The seed of the pseudo random number generator to use when shuffling the data
- tol = 0.0001. Tolerance for stopping criteria
- warm_start = False. When set to True, reuse the solution of the previous call to fit as initialization, otherwise, just erase the previous solution

The last step is to train created Machine Learning models in order to classify websites. This is done by passing generated training features set and training labels set to the fit method. Fitting models to the training data is essentially the training part of the modeling process. It finds the coe cients for the equation specified via the algorithm being used.

Machine learning models after fitting procedure could predict category of the website. The results of prediction may concern of how well data was prepared and also what kind of Machine Learning algorithm was chosen. The prediction is done by using Supervised learning predict function. This function input is test features set and it returns a labels set of a given features.

3.4.3 Models performance evaluation

Models performance evaluation score is a result of how well models are trained. Evaluation process is performed using 2 di erent data sets with di erent websites list. First data set is the original data set from where training/testing features/labels set were created. Second data set is custom made which contains english language websites links with labeled categories.

1. **Models performance evaluation using original data set**.

   Primary data set allows evaluate performance for three di erent parameters: Precision score, Recall score and F1-score. Scores evaluations are made for 2 models: Linear Regression model and Linear Support Vector Machine model.

   Since models solves multi-class machine learning approach, because labels consist of 25 di erent categories, evaluations methods are applied on each category.

Logistic Regression model prediction evaluation analysis:

	Logistic Regression		
Category	**precision**	**recall**	**f1-score**
Recreation_and_Hobbies	0.98	0.86	0.91
Food_and_Drink	0.84	0.86	0.85
News_and_Media	0.86	0.92	0.89
Travel	0.87	0.66	0.75
Career_and_Education	0.93	0.80	0.86
Home_and_Garden	0.88	0.86	0.87
Internet_and_Telecom	0.87	0.81	0.84
Gambling	0.80	0.90	0.85
Books_and_Literature	0.87	0.95	0.91
Science	0.95	0.88	0.91
Health	0.82	0.59	0.69
Autos_and_Vehicles	0.94	0.89	0.92
Finance	0.80	0.86	0.83
Sports	0.87	0.85	0.86
Adult	0.81	0.83	0.82
Pets_and_Animals	0.92	0.90	0.91
Business_and_Industry	0.80	0.90	0.85
People_and_Society	0.83	0.26	0.40
Law_and_Government	0.89	0.87	0.88
Beauty_and_Fitness	0.94	0.91	0.93
Arts_and_Entertainment	0.76	0.79	0.78
Games	0.88	0.81	0.84
Reference	0.85	0.76	0.80
Computer_and_Electronics	0.25	0.11	0.15
Shopping	0.94	0.85	0.89

Table 3. Logistic regression model performance evaluation for each category

Logistic regression model evaluation scores table (3) indicates that the most positive predictions (*precision score*) model made on "Recreation_and_Hobbies" category with **0.98 precision score**; category of the highest actual positives identification proportion (*recall score*) is "Books_and_Literature" with **0.95 recall score**; category of the highest harmonic mean of precision and recall scores is "Beauty_and_Fitness" with **0.93 f1 score**.

Logistic Regression model overall mean scores for all categories:

- Accuracy score: 0.85
- Precision score: 0.85
- Recall score: 0.79
- F1 score: 0.81

Logistic Regression model confusion matrix heat map for each category (17):

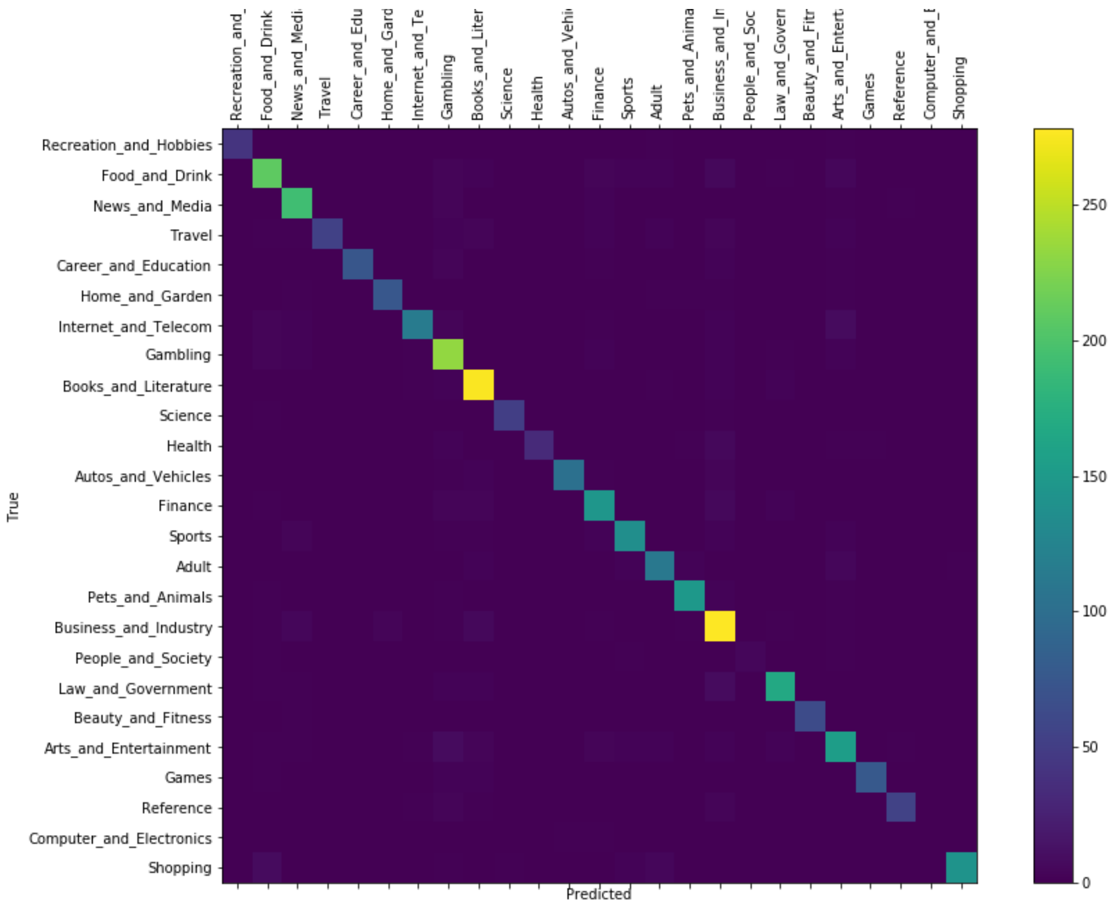


Figure 17. Confusion matrix of Logistic Regression model

Linear Support Vector Machine model prediction evaluation analysis:

	Linear Support Vector Machine		
Category	**precision**	**recall**	**f1-score**
Recreation_and_Hobbies	0.78	0.80	0.79
Food_and_Drink	0.81	0.78	0.79
News_and_Media	0.81	0.87	0.84
Travel	0.79	0.54	0.64
Career_and_Education	0.83	0.72	0.77
Home_and_Garden	0.79	0.84	0.8
Internet_and_Telecom	0.76	0.72	0.74
Gambling	0.76	0.83	0.79
Books_and_Literature	0.86	0.88	0.87
Science	0.86	0.86	0.86
Health	0.62	0.52	0.56
Autos_and_Vehicles	0.86	0.84	0.85
Finance	0.76	0.78	0.77
Sports	0.73	0.75	0.74
Adult	0.77	0.75	0.76
Pets_and_Animals	0.88	0.88	0.88
Business_and_Industry	0.79	0.85	0.82
People_and_Society	0.36	0.21	0.27
Law_and_Government	0.82	0.82	0.82
Beauty_and_Fitness	0.86	0.78	0.82
Arts_and_Entertainment	0.72	0.74	0.73
Games	0.71	0.73	0.72
Reference	0.75	0.65	0.70
Computer_and_Electronics	0.08	0.11	0.09
Shopping	0.81	0.77	0.79

Table 4. Linear support vector machine model performance evaluation for each category

Linear support vector machine model evaluation scores table (3) indicates that the most positive predictions (*precision score*) model made on "Pets_and_Animals" category with **0.88 precision score**; category of the highest actual positives identification proportion (*recall score*) is "Books_and_Literature" and "Pets_and_Animals" with **0.88 recall score**; category of the highest harmonic mean of precision and recall scores is "Pets_and_Animals" with **0.88 f1 score**.

Linear support vector machine model overall mean scores for all categories:

- Accuracy score: 0.79
- Precision score: 0.74
- Recall score: 0.72
- F1 score: 0.73

Linear support vector machine model confusion matrix heat map for each category (18):

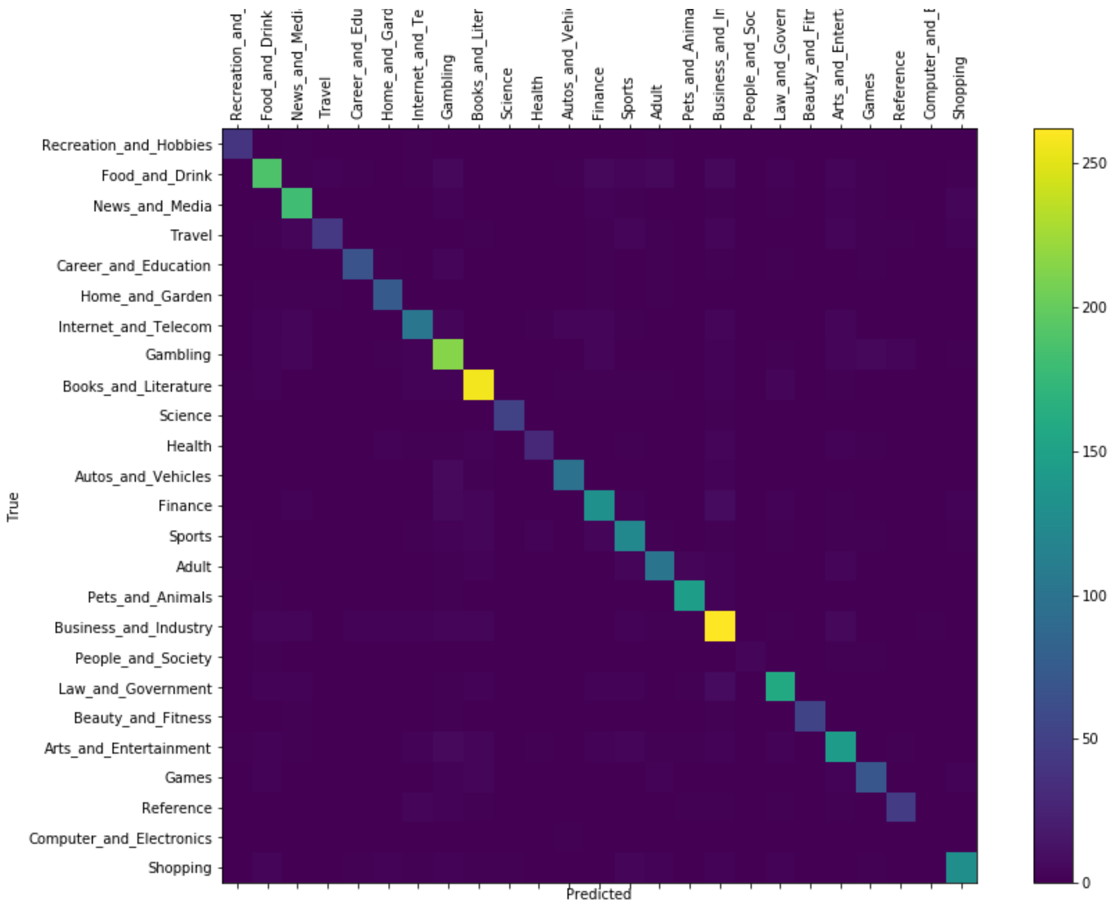


Figure 18. Confusion matrix of Linear Support Vector Machine model

Models performance overall evaluation (1, 1) results implies that model with the best performance for classifying websites categories is **Linear Regression** model.

2. **Models performance evaluation using custom made data set**.

   Original data set evaluation score is not an final indicator that machine learning models are performing well. The original data set has two serious flaws which could have an impact for final models results.

   (a) Data set could be outdated. The original data set is created in 2014 year so some of websites content could be changed or not working that leads to another problem that data set classified labels could be changed in the current time.

   (b) Classified labels could be wrong. The original data set contains 31086 di erent websites so to check each website category would consume too much time. Revision of labeled websites would be great to make sure that declared category of particular website is correct

   These problems are solved by creating small custom made data set for testing performance on created machine learning models. The custom data set consist of 288 working english websites with human confirmed categories. Each website is iterated and content is download in the same manner of original data set.

On custom model data set is tested my created website categorization model **??** which predicts category of website by creating feature set for each category and calculating weight of the features set. The feature set with the highest weight is considered to be category of the website.

Performance evaluation using custom made data set on my custom model and machine learning models, testing on 286 di erent websites and there are performance results:

- **Custom model**: 69.5 % (197 correct predictions of 282)
- **Logistic regression model**: 57.8 % (163 correct predictions of 282)
- **Linear support vector machine model**: 52.5 % (148 correct predictions of 282)

From given results custom model performance is the best for predicting up to date websites categories. Machine learning models performance is similar - logistic regression manage correctly predict 163 websites categories while linear support vector machine 148 websites categories. According to original data set machine learning models results - linear regression model should perform better than linear support vector machine and that confirms performance results on the custom data set.

Models performance on predicting each category:

	Custom data set performance evaluation		
Category	**Custom model**	**LR model**	**LSVM model**
Food_and_Drink	6 / 7 : 85.71 %	6 / 7 : 85.71 %	6 / 7 : 85.71 %
News_and_Media	12 / 14 : 85.71 %	11 / 14 : 78.57 %	11 / 14 : 78.57 %
Travel	13 / 15 : 86.67 %	8 / 15 : 53.33 %	7 / 15 : 46.67 %
Career_and_Education	14 / 14 : 100.00 %	11 / 14 : 78.57 %	9 / 14 : 64.29 %
Home_and_Garden	3 / 16 : 18.75 %	2 / 16 : 12.50 %	2 / 16 : 12.50 %
Internet_and_Telecom	3 / 16 : 18.75 %	3 / 16 : 18.75 %	2 / 16 : 12.50 %
Gambling	9 / 9 : 100.00 %	6 / 9 : 66.67 %	5 / 9 : 55.56 %
Books_and_Literature	7 / 10 : 70.00 %	7 / 10 : 70.00 %	7 / 10 : 70.00 %
Science	16 / 20 : 80.00 %	13 / 20 : 65.00 %	11 / 20 : 55.00 %
Health	9 / 11 : 81.82 %	6 / 11 : 54.55 %	6 / 11 : 54.55 %
Autos_and_Vehicles	13 / 16 : 81.25 %	13 / 16 : 81.25 %	12 / 16 : 75.00 %
Finance	13 / 19 : 68.42 %	12 / 19 : 63.16 %	12 / 19 : 63.16 %
Sports	8 / 9 : 88.89 %	7 / 9 : 77.78 %	5 / 9 : 55.56 %
Adult	8 / 8 : 100.00 %	7 / 8 : 87.50 %	5 / 8 : 62.50 %
Pets_and_Animals	9 / 12 : 75.00 %	5 / 12 : 41.67 %	5 / 12 : 41.67 %
Business_and_Industry	4 / 8 : 50.00 %	3 / 8 : 37.50 %	3 / 8 : 37.50 %
People_and_Society	1 / 1 : 100.00 %	1 / 1 : 100.00 %	1 / 1 : 100.00 %
Law_and_Government	6 / 8 : 75.00 %	4 / 8 : 50.00 %	4 / 8 : 50.00 %
Beauty_and_Fitness	19 / 23 : 82.61 %	16 / 23 : 69.57 %	14 / 23 : 60.87 %
Arts_and_Entertainment	2 / 10 : 20.00 %	2 / 10 : 20.00 %	2 / 10 : 20.00 %
Games	3 / 6 : 50.00 %	2 / 6 : 33.33 %	2 / 6 : 33.33 %
Reference	1 / 10 : 10.00 %	1 / 10 : 10.00 %	1 / 10 : 10.00 %
Computer_and_Electronics	12 / 13 : 92.31 %	12 / 13 : 92.31 %	12 / 13 : 92.31 %
Shopping	6 / 7 : 85.71 %	5 / 7 : 71.43 %	4 / 7 : 57.14 %

Table 5. Models performance on predicting each category

According to the custom data set performance results(5) models performance on predicting category the best results are performed on category "Career_and_Education" where custom model predicts of 100 % accuracy; logistic regression model and linear support vector machine model predict on 78.57 % accuracy. Not all predictions on categories are equal, because there are different websites instances of categories. For example: category "People_and_Society" all models predicted at 100 % accuracy, but this is not quite accurate since there is only 1 website with in the custom data set with that category. To make this evaluation more accurate, the custom data set should be expanded with more websites instances and websites instances on categories number should be equal for all categories.

Conclusions and Recommendations

All created models accuracy on predicting real world websites categories are greater than 50 %. Statistically that models most likely should have more correct predictions than incorrect in a long term.

During implementation part there was created 3 models which are capable of classifying websites:

1. **Custom model**
2. **Logistic Regression model**
3. **Linear Support Vector Machine model**

There are 2 websites data sets with websites URL's and categories:

1. **Original data set**

   Original data set is an open source and it is available at Data For Everyone data set lists. The data set contains 31086 di erent URLs with 81 attributes.

   This data set is used to create *the most frequent words list for categories* and *training/testing features and labels sets* for machine learning models.

   Machine learning performance on the original data set features set:

   (a) **Logistic Regression**
   - Accuracy score: 0.85
   - Precision score: 0.85
   - Recall score: 0.79
   - F1 score: 0.81

   (b) **Linear Support Vector Machine**
   - Accuracy score: 0.79
   - Precision score: 0.74
   - Recall score: 0.72
   - F1 score: 0.73

2. Custom data set

   Custom data set is human made website data set with 282 di erently labeled websites.

   Models accuracy performance on the custom data set:

   (a) **Custom model**: 69.5 % (197 correct predictions of 282 websites)

   (b) **Logistic Regression**: 57.8 % (163 correct predictions of 282 websites)

   (c) **Linear Support Vector Machine**: 52.5 % (148 correct predictions of 282 websites)

List of methods that would improve project results:

- **Download content of website recursively**.

  This would take a lot of time but may improve the most words frequency list for categories generation since all links in the website would be visited and all information would be scraped.

  At this project just one website home page is scraped and it takes about 23 hours to analyse all websites from the original data sets. Scraping all website links would take much more time but it would be more e cient

- **Revise website categories in the original data set**.

  Original data set contains 31086 instances of labeled websites. However the data set was created in 2014 year so it is possible that websites content is changed and the category is also changed according to the website content. The solution would be to have human validation on website categories to confirm if declared category match the reality and is valid.

- **Translate non english content websites to english language**.

  Models are capable to predict categories for those websites which the language of content is english. This is the problem because the number of websites is reduced since there are a lot of websites with non english content. In the original data set after filtering out non english websites the number of websites instances reduced from 31086 websites to 10436 websites.

  The solution would be translate website non english content to the english language. However, at the moment only "Google" are o ering translation API services but these services are with symbols and intensity restrictions which do not allow properly translate websites.

  Translation implementation would allow models to predict any website with any language and automatically the variaty of websites would be much bigger.

- **Improve custom data set** .

  1. The custom data set should be expanded with more websites instances
  2. Websites instances on categories number should be equal for all categories. Some categories do not have any website or have just a few websites

- **Revise categories**.

  Categories of websites sometimes could not be identified accurately since the website data could contain words that would cover multiple categories. Some categories are similar to other categories, for example: "Business_and_Industry" and "Finance" or "Internet_and_Telecom" and "People_and_Society".

  For each category need to make an analysis to define main key words for particular category.

References

[1] http://onlinestatbook.com/2/regression/intro.html.

[2] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995.
http://home.ku.edu.tr/mehyilmaz/public_html/mean-shift/00400568.pdf.

[3] Ian Davidson and SS Ravi. Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 59–70. Springer, 2005.
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.439.1543&rep=rep1&type=pdf.

[4] Tom Fawcett Foster Provost. *Data Science for Business*. 2013.
http://shop.oreilly.com/product/0636920028918.do.

[5] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001.
https://web.stanford.edu/~hastie/ElemStatLearn/printings/ESLII_print12.pdf.

[6] Yoav Goldberg. *Neural Network Methods for Natural Language Processing*. 2017.
https://www.morganclaypool.com/doi/abs/10.2200/S00762ED1V01Y201703HLT037.

[7] Joel Grus. *Data science from scratch: first principles with python*. " O'Reilly Media, Inc.", 2015.
http://shop.oreilly.com/product/0636920033400.do.

[8] David Lane. Introduction to linear regression.
http://onlinestatbook.com/2/regression/intro.html.

[9] David M Magerman. Statistical decision-tree models for parsing. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 276–283. Association for Computational Linguistics, 1995.
http://www.aclweb.org/anthology/P95-1037.

[10] Kishore Papineni. Why inverse document frequency? In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics, 2001.

[11] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM New York, 2001.
https://www.cc.gatech.edu/~isbell/reading/papers/Rish.pdf.

[12] Darshan Sonagara and Soham Badheka. Comparison of basic clustering algorithms. *Int. J. Comput. Sci. Mob. Comput*, 3(10):58–61, 2014.
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.439.1543&rep=rep1&type=pdf.

[13] Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, and Vipin Kumar. *Introduction to Data Mining. Second Edition*. Pearson Education (US), 2018.
https://www-users.cs.umn.edu/~kumar001/dmbook/index.php.

[14] Laurens van der Maaten, Eric Postma, and Jaap van den Herik. Dimensionality reduction: A comparative review. Technical Report TR 2009–005, TiCC, Tilburg University, Tilburg centre for Creative Computing, 2009.
http://www.math.chalmers.se/Stat/Grundutb/GU/MSA220/S18/DimRed2.pdf.

[15] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.
http://files.isec.pt/DOCUMENTOS/SERVICOS/BIBLIO/Documentos%20de%20acesso%20remoto/Principal%20components%20analysis.pdf.

[16] Jieping Ye, Ravi Janardan, and Qi Li. Two-dimensional linear discriminant analysis. In *Advances in neural information processing systems*, pages 1569–1576, 2005.
http://papers.nips.cc/paper/2547-two-dimensional-linear-discriminant-analysis.pdf.