Open University of Cyprus

Faculty of Pure and Applied Sciences

Master's Degree

Information and Communication Systems

Web Data Analysis and Classification: Automated Text Classification by Linguistic Norms, Content and Genre

Christos Paschalidis

Supervisor
Dr Chrystalla Neofytou

November 2020

Open University of Cyprus

Faculty of Pure and Applied Sciences

Web Data Analysis and Classification: Automated Text Classification by Linguistic Norms, Content and Genre

Christos Paschalidis

Supervisor
Dr Chrystalla Neofytou

This dissertation was submitted in partial fulfilment of the requirements

for obtaining a Master's Degree

in Information Systems

from the Faculty of Pure and Applied Sciences
of the Open University of Cyprus

November 2020

Abstract

The dissertation aimed to expand the effort of utilizing the largest collection of texts on the Internet. The purpose of this work was to critically approach the methods of analysis and classification of web data and the creation of the deliverable system (Katigoriopoiitis) that utilizes linguistic norms, content and genre of websites in order to facilitate the way in which this data is presented. A bibliographical research on Web Data Mining aimed to describe the techniques of collecting information from the web. A presentation and cross comparison of machine learning algorithms (Naïve Bays, Decision Trees, K-Nearest neighbors and Support Vector machines) aimed to find the best fit for general purpose content classification for the implementation of the classifier. Accuracies of different classification models were tested on the same dataset. The outcome of the dissertation was that there are efficient techniques that can be applied in order to sufficiently use Internet information. Internet technologies and standards are getting richer and this maximizes the options for data mining. Content classification can be easily achieved by using simple model implementations. More sophisticated models are needed in order to achieve high accuracy in sentiment analysis, or classification based on linguistic norms.

Keywords: Machine Learning, Natural Language Processing, NLP, Web Data Mining, Text Classification, Internet, World Wide Web

Acknowledgements

I am thankful to Milica Miljanic for helping me organizing this essay and constantly reminding me the importance of the research. Without her, this task would be remained incomplete. I would also like to thank Marko Markovic, Milena Petrovic, Jelena Bojic and Davor Pancic for their honest feedback.

A special thanks to my supervisor Dr Chrystalla Neofytou for being helpful and patient with my limited time due to professional obstacles.

Table of Contents

Chapter 1
Introduction

The information that can be found scattered on the Internet is impossible to be measured accurately, while the quality of its utilization is becoming more and more difficult. Considering the amounts of unstructured data on the Internet and their rapid growth, inventing automated tools that can effectively organize these texts becomes more than a necessity. This dissertation is focusing on finding efficient techniques and algorithms in order to create a tool (Katigoriopoiitis) which will be able to classify web texts based on their linguistic norms, content and genre.

As Bowerman noted in 2006, "traditionally, a language is thought to be structured along a set of rules, or 'norms,' that prevail over all aspects of the language: phonology, morphology, syntax, and semantics. These norms serve to make the language distinctive, intelligible within a wide speech community, and learnable. However, the precise definition of a norm is controversial and difficult. It is important to distinguish between those norms (which I shall call 'descriptive norms') that enable us to describe a language or variety from observation of data, and prescriptive or 'pedagogical norms,' which often reflect some abstract ideal of how a language should be used, rather than the actual practice of native speakers of that language" [1]. Taking into consideration this separation of the language norms, we can presume that: by knowing all the 'pedagogical

norms' of a language, we are able to apply just a formal set of rules and not to completely understand or reproduce all the variations of the language. Furthermore, a language can have variations based on geographical location, context, age of the speaker/writer, and form. For example, the language that is used in a court, can be far different from the language used in a school, or a radio station. Native speakers learning the 'descriptive norms' based on their observation and experience. In the present dissertation techniques, algorithms and classification models, designed and trained in a variety of different contexts are presented, in order to be able to classify texts. One of the aspects of this training process, is to find patterns such as norms that can be observed by processing huge datasets rather than described by concrete rules. Content classification without taking into consideration the norms of a language[1], can achieve good accuracy but this disability of utilizing the norms makes classification context insensitive. While content can be classified with or without context, genre classification is way more complicated. "Genre is necessarily a heterogeneous classificatory principle, which is based among other things on the way a text was created, the way it is distributed, the register of language it uses, and the kind of audience it is addressed to" according to Brett Kessler, Geoffrey Nunberg and Hinrich Schutze [2]. Genre classification can give an additional context for content classification. Genre detection has a lot of aspects. For example, poems have a variety of structural characteristics such as thyme, rhythm, alliteration, personification that makes them different than novels; however, the fact that a text can have extensive usage of metaphors or personification or even rhyme, does not make it a poem. On Internet genre categorization can for example be the separation of the electronic newspapers and social media texts. A social media post can be shorter, using special or sub-language, extensive usage of punctuation marks such as hashtags[2] etcetera.

This dissertation is making a bibliographic review of the topic of analysis and classification of web data, followed by a bibliographic and comparative study of automatic text sorting systems. Additionally, machine learning algorithms are compared in Python using an English dataset. Training and experimental measurement of the text classifier performance results can be found in the appendices. The innovative aspect of the research will be the text classifier, which will be the deliverable of this dissertation.

[1] See the simple Bag of Words implementation.
[2] Hashtag (#) is extensively used in social media to mark the content with keywords or key phrases related with a general topic, condition or event for example: #summer_vacations, #elections2020 #covid19 etcetera.

1.1 Approach to the Problem

Internet[3] data can be unstructured, unconfirmed and chaotic. Also, it is hard to identify the authors of the web content and this raises questions about the authenticity of the content itself. If the data is not authentic, it can provide wrong information and be misleading. For example, text classification algorithms can be applied in order to evaluate the authenticity of the reviews in E-commerce platforms. According to Saleh Nagi Alsubari, Mahesh B. Shelke and Sachin N. Deshmukh, "fake reviews posted in Ecommerce websites represent opinions of customers in which these reviews play a crucial role in e-business because they can indirectly affect future buying decisions" [3].

The main reasons of the unstructured nature of the Internet can be found in the fast growth of the technologies around it, as well as in its open nature. The great majority of the texts on the Internet is stored in HTML[4]. While HTML documents created in order to provide a fast and universal way to access (mainly) text data[5], it took some time to encapsulate information that help on classification such as metadata keywords and text semantics. This technical gap caused the early information on the Internet to be even harder to extract information due to multilinguality of semantic Web technologies [4]. Additionally, since anyone is able to create and publish websites, it is not possible to control the way the information is presented, whether it has metadata and semantic information that describes the content accurately.

The main and dominant way to access information on Internet, is via search engines. During years, search engines developed advanced Machine Learning techniques to organize the constantly increasing Web Data. Search engines are using bots called Crawlers that navigate to the hyperlinked pages, collecting metadata and semantics, mapping the websites and building a searchable dictionary of the entire Web [5]. The purpose of Data Mining and Machine Learning, as it will be presented to the third chapter of this dissertation, is to find patterns in a large scale of

[3] Often when Internet is mentioned, it refers to the World Wide Web (WWW). For clarity, it is important to separate these two terms. The first, is the global computer network that gives the ability to interconnect billions of devices all over the world using the Internet Protocol [61]. The World Wide Web proposed by Sir Tim Berners-Lee and Robert Cailliau in 1990 as a "universal hypertext system" [60]. The Web is worldwide thanks to the Internet. Despite the fact the Internet and the WWW are obviously two distinct terms, in this dissertation for simplicity and readability, the Internet is mentioned in a broader meaning that mainly refers to the World Wide Web.

[4] HTML: Short of Hypertext Markup Language.

[5] In 1989 Sir Tim Berners-Lee proposed an information management system based in hypertext [28] with main goal to find a better way to organize, link and share scientific documents in CERN. This idea was the base of the World Wide Web.

raw data, in order to be able to make decisions and extract useful information. But is it possible to classify this amount of information using these techniques? The sufficient use of the information on Internet is questionable.

1.2 Contribution

There is plenty of tools and services aiming for utilizing the Web Data. This dissertation presents some of the most popular scientific and development tools for text analysis and classification [6] [7] and continues with tools for text extraction. The deliverable tool of this dissertation (Katigoriopoiitis), aims to add additional value by introducing a SaaS[6] solution for Web text classification. Katigoriopoiitis is mainly inspired by WebCorp tool and is able to classify Web search results in real time. Additionally, it is using techniques for clean content and metadata extraction. The tool can be trained in multiple topics and languages by feeding labeled texts. Furthermore, multiple predictions are possible[7]. The tool can be easily integrated with other services, web pages or mobile applications since its API is implemented over HTTP.

1.3 Key research questions

The key questions this dissertation aims to answer can be concluded as follows:

- Can the information on the Internet be sufficiently used?
- In which ways can it be classified?
- By which methods can we classify texts based on their linguistic norms, the content and the genre?

[6] SaaS: Short of Software as a Service [63]
[7] For example, a model can be trained to detect the text Genre (such as news, literature, poetry) while a second prediction model can be trained in order to label the content (politics, sports, tech etcetera).

Chapter 2
Methodology

A bibliographic research was implemented with publications, articles and works related to the field. The dissertation presents existing tools for text classification and web content extraction. This was followed by a bibliographic (and comparative) study of automatic text classification systems with techniques from various scientific fields (decision trees, Naive Bayes, Nearest Neighbor, Support Vector Machines, Metadata, Ontologies). Following the scientific and technical research, the deliverable text classifier specifications were created. As it was presented in third chapter of this dissertation, a variety of scientific and development tools are already covering the topic of text classification and extraction however, there is still space of improvement in web search classification. Katigoriopoiitis is designed to be easy to use from users that are not familiar with NLP techniques, as well as for experienced users. All the techniques and the algorithms implemented in the tool are extensively presented in the dissertation.

2.1 Tools

The main tool of the dissertation was chosen to be **Python** programming language due to the variety of machine learning libraries that offers. Text classifier API was written in **Django** framework. The main Python libraries used were: **nltk**, **sklearn** and **pandas**.

The criteria of using those libraries were:

- To have independency of third-party services as much as possible
- To be well documented
- To be simple to use and extendable

NLTK [8] offers tools for text preprocessing such as for stemming and lemmatization, supporting several languages, and tokenization functions. Additionally, it offers a library of corpora that can be used for testing purposes. It mainly used for the Stemming tools.

Greek Stemmer [9] was used for stemming Greek texts since NLTK does not support it.

Scikit-Learn has implemented plenty of algorithms for supervised machine learning, as well as tool. It was used for training and using text classification machine learning models.

Git [10] was used for version control and the code has been stored on **GitHub** [11].

Heroku [12] cloud service was used for publishing the Classifier API. The reason is that it offers a free plan and also it is integrated with GitHub.

Postman [13] was chosen for API testing.

Figma [14] and was used for creating the Figures.

2.2 Web Text Classifier (Katigoriopoiitis)

A Python project was created based on the research of the dissertation. Classifier designed as an API service. Which means it can be easily accessible from third-party software and technologies. The classifier was tested using Postman. An extensive presentation of the methods of analysis and classification of web data (Data Mining techniques, Machine Learning, Semantic Web) was done in order to explain how the tool works. Classifier has been designed in order to be flexible with data input. The tool supports three popular file formats:

- Comma-separated values (csv)
- Excel spreadsheets (xlsx)
- JavaScript Object Notation files (json)

2.3 Testing classification algorithms

Classifier is designed to be a rapid, real-time and easy to use tool. In contrast to other tools, it is flexible in classification by giving the ability to the user to train and use classification models easily. Models with different implementations were created and tested in Python in order to specify which model achieves the best accuracy in text classification. Based on these, model presets have been created, in order users with limited knowledge on the field to be able to achieve high accuracy, without the need of tweaking the model settings. The training set had "2225 documents from the BBC news website corresponding to stories in five topical areas from 2004-2005. Class Labels: 5 (business, entertainment, politics, sport, tech)" [15].

Chapter 3
Web Text Classification Tools

We are living in the era that analysis of web data is present in business, science, marketing and almost all other fields of human productivity. Data mining and machine learning tools that can be used by both machine learning experts and none experts are becoming more and more popular and necessity [16]. On the other hand, World Wide Web is enormously big source of data available to billions of people [17].

Data mining is a very broad field of data sciences that inherits its techniques from variety of fields. There is a common confusion on how is data mining related with machine learning and statistical modeling. A recent definition of data mining can be found in a publication of David J. Hand and Niall M. Adams: "Data mining is the discovery of structures and patterns in large and complex data sets. There are two aspects to data mining: model building and pattern detection. Model building in data mining is very similar to statistical modeling, although new problems arise because of the large sizes of the data sets and the fact that data mining is often secondary data analysis. Pattern detection seeks anomalies or small local structures in data, with the vast mass of the data being irrelevant." [18].

A lot of similarities in techniques applied in Machine Learning can be found in Data Mining. According to Tom Mitchell: “In the field known as data mining, machine learning algorithms are being used routinely to discover valuable knowledge from large commercial databases containing equipment maintenance records, loan applications, financial transactions, medical records, and the like” [19]. As a result, machine learning can be considered as a part of data mining field.

Arthur Samuel described machine learning in 1959 as: "the field of study that gives computers the ability to learn without being explicitly programmed" [20]. This is an older, informal definition.

Tom Mitchell’s definition from 1997 provides a more modern assumption: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" [21].

Machine Learning can be separated in two main types:

- **Supervised Machine Learning**

- **Unsupervised Machine Learning**

In supervised learning [22], the training set is labeled. In text classification for example, when a model is trained using supervised learning, the training set is already tagged with class labels (the sports content will have a tag **sport**, the political content **politics** etcetera). The purpose of training is to “teach” the model, which content matches with which label. The idea is that there is a relationship between the input and the output in training data set. That’s why the output is considered as dependent value from independent input features in a data set.

Supervised learning problems are divided into two categories:

- **Regression**: Problem prediction results are within a continuous output. It means that input data is mapped to continuous numerical output.

- **Classification**: Results are predicted in discrete categories.

The unsupervised learning [23] does not need a labeled training dataset, and its algorithms can be divided in two categories:

- **Clustering**: These algorithms are good in finding patterns in order to split the training data into groups. It can be used for classification of unknown datasets. For example, a language agnostic text classification model can be built using Clustering algorithms.

- **Association**: These algorithms are aiming to find patterns in order to associate related subjects. For example, it can find the association between e-commerce buyers and make predictions for product recommendations.

Text classification is obviously classification problem. It can be treated both as unsupervised and supervised learning problem, depends on purpose of the task. This dissertation is focusing on supervised text classification techniques and most popular algorithms for supervised classification.

There is variety of tools that can allow us accessing or processing of web data. In this chapter some of the most popular and modern tools in the field[8] will be presented. The main goal is to present what are the possibilities at the time aiming to find a potential space of improvement.

3.1 Tools for document classification

Apache OpenNLP is a Java based advanced tool offering text classification through its Document Classifier (Doccat) for training models. It incorporates tools for tokenization, sentence segmentation, named entity extraction, and language detection. Models created in OpenNLP can be used through its coding API. Despite the fact it is originally written for Java, there are wrappers for other languages. It is open source and it needs to be downloaded and installed [24].

Orange Data Mining is another open source data mining tool. Its sophisticated interface makes it suitable for educational purposes as well as for scientific purposes. Orange Data Mining offers tools for text classification, sentiment analysis, text clustering, data preprocessing, popular data set analyses (twitter). It offers workflows that can be easily used from non-developer users, but advanced users have the ability to write Python snippets for more complex tasks. Additionally, it can be used as a Python library [25]. Other features are data visualization, cross validation tests and misclassifications. It can be downloaded and installed on Windows, Mac and Linux [26].

[8] Jan Pomikalek presents and compares some of those tools in a PhD thesis in 2011. The thesis approaches the limitations of the Web search engines and aims to solve the problem of data extraction and duplication [66].

DatumBox is another Java based tool. The difference with the tools mentioned above is the fact that it is offered as SaaS[9] as well as a framework that can be downloaded and used locally. It its toolset contains sentiment analysis, twitter sentiment analysis, topic classification, keyword extraction, subjectivity analysis, adult content detection and spam detection. DatumBox's topic classification function assigns documents in 12 (predefined) thematic categories [27].

Google Cloud offers tools for text, image and voice classification. Its NLP API contains plenty text classification and analysis tools. Sentiment analysis, syntax segmentation, entities and entity sentiments as well as content classification are the main features of the API. Google Cloud gives plenty of general purpose pretrained models and gives the option for custom data training. It is a full-featured commercial service that can be integrated in many programming environments [28].

3.2 Web Corpora Tools

WebCorp is an online tool for utilizing web texts as corpora. It is a real-time tool powered by Bing search engine and The Guardian Open Platform [29]. The main tool is a search engine with a free query string and additionally options of span size and case sensitivity. It also offers a wordlist tool for that generates word frequencies of a URL content [30] [31].

BootCat is a free tool written in Java, that bootstraps web texts to corpora. It builds the corpora based on input terms and stores the crawled pages in the local file system. It offers command line tools that can be used for automating the crawling process or third-party tools. BootCat needs to be downloaded and can be installed on Windows, Mac and Linux. Its source code is available on GitHub [32].

SketchEngine is a commercial online corpora creator. In its tool set has word collections, term extraction and bilingual term extraction, word frequency tools, morphological analysis and part of speech tagging. It also performs neologisms and diachronic analysis of word usage. Its crawling tool is based on BootCat, and it has the benefit of running as a service on cloud instead of local machine [33].

[9] SaaS: Short of Software as a Service [63]

3.3 Web Content cleaning Tools

The content of Web pages is surrounded by destructive information such as navigational elements, template markup and advertisements. Cleaning the web content is necessary for readability purposes for example the **Instapaper** [34] and the Arc90's **Readability**[10] tools as well as the browser AdBlocking[11] tools. Another important reason of cleaning web content, is for text classification analysis. Web texts need to be clean of unrelated information in order to be classified.

Figure 1: Content in this web page is surrounded by destructive information.

As can be seen in **Figure 1**, the useful information of the article is the **Title**, the **Author**, the **Date of publication** and the **Actual Content** of the article, while the rest of the items surrounding this

[10] The official Readability is not active anymore. Forks of the source code and its webpage can be still found [64].
[11] Mshabab Alrizah, Sencun Zhu, Xinyu Xing and Gang Wang analyzing in 2019, the challenges of cleaning advertisement content (ads) in AdBlocking software [65].

information is destructive and not related to the article. Despite the fact that it can be relatively easy for a human to locate the related information. Before text classification algorithms take place, content needs to be clean in order to be sufficiently classified.

3.4 Space of improvement

Considering the existing toolbox for Web data mining and text classification, a lot of important work has been done[12] [35]. Focusing on the instant tools such as the web search engines, and SaaS APIs, a generic tool that can be continuously trained in different (custom) domains, would add an additional value. WebCorp is a great example of a real-time tool, however its search is limited in a very short character span[13]. Another limitation is that it does not offer content classification. DatumBox offers a service with plenty of tools, however, the content classification is limited in a general purpose of 12 predefined thematic categories. By using Apache NLP or Orange, online tools can be implemented, however, there is no real-time service offered out of the box. Furthermore, Google Cloud can be easier to be used as a base for creating a real-time service, however, it needs some development effort in order to satisfy the needs of a real-time Web classifier.

[12] It is important to be mentioned that plenty of advanced linguistic tools can be found. This dissertation focuses on finding solution in Web classification problem and cannot review all the important work that has been done in the topic.
[13] WebCorp returns results of 100 characters or 10 words span.

Chapter 4
Classification Algorithms

The purpose of this chapter is to present some of the algorithms that can be used for text classification. The following algorithms are based in different techniques. The size of the dataset and the nature of the task are crucial during the selection of the algorithm. Different algorithms have different pross and cons and based in the nature of the data they are applied to, can give better or worse results in comparison with others. An experiment including implementations of classification models based on Naïve Bayes, Decision Trees, K-Nearest Neighbors and Support Vector Machine algorithms was executed during the research and the results can be found in the **Appendix A**.

4.1 Naïve Bayes

The **Naïve Bayesian** algorithm is simple probabilistic algorithm based on Bayesian theorem. It's called naïve because it considers independence among the features, or that effect of one feature on prediction of a class is independent from values of other features.

It is based on Bayesian formula:

$$\mathbf{P(y|x) = \frac{P(x|y)P(y)}{P(x)}}$$

P(y|x) is the probability of target class **y**, given attribute **x**

P(x|y) is the probability of attribute **x**, given class **y**

P(x) is the probability of **x**

P(y) is probability of class **y**

In case of a text sentiment analyses or binary classification, the attribute **x** is the token (word) from corpora and **y** is the target class that can be positive or negative. First step in Naïve Bayesian is to make table of probabilities of all attributes in the given classes.

$$P(\mathrm{w_i}|\mathrm{class}) = \frac{\mathrm{freq}(\mathrm{w_i}, \mathrm{class})}{\mathrm{N_{class}}}$$

freq(w_i, class) is the number of words $\mathbf{w_i}$ in the given class

$\mathbf{N_{class}}$ is the number (frequency) of all words in the given class

The ratio of all positive and negative training samples (prior ratio), is useful to avoid unbalanced data. The ratio of conditional probabilities is used to estimate likelihood of an attribute in target class. Product of a primer ratio and all sample likelihoods gives Bayesian inference that computes the posterior probability according to Bayes' theorem:

$$\frac{P(pos)}{P(neg)} \prod_{i=1}^{m} \frac{P(w_i|pos)}{P(w_i|neg)} > 1$$

Where **m**, is the number of words in new test sample, and probability of a class is computed as a product of ratio of conditional probabilities of all words in that sample to be in positive or negative class. Decision boundary in this binary classification example is set at **1**. If the likelihood is greater

than **1**, it is positive class and otherwise negative in binary classification. If it is equal to **1**, it is considered as neutral.

It's obvious that there is a problem if words from new test sample do not appear in a vocabulary (data set) or in a specific class. It would annulate all other probabilities and make product equal to zero. In order to avoid probabilities being zero there is technique called Laplacian smoothening. Instead of counting conditional probabilities as a frequency of a word given class divided with the number of all words in a class, there is a slightly different formula that adds **1** to nominator and **V** (number of unique words in corpus) to denominator:

$$P(\mathrm{w_i}|\mathrm{class}) = \frac{\mathrm{freq}(\mathrm{w_i}, \mathrm{class}) + 1}{\mathrm{N_{class}} + \mathrm{V}}$$

Another issue is that the when we are computing probabilities, the product can be very small, and will quickly go to zero to the numerical precision of floating-point numbers. To resolve this issue, a simple solution is to instead compute the probability in their log space with decision boundary set at 0:

$$\log\left(\frac{P(pos)}{P(neg)}\prod_{i=1}^{n}\frac{P(w_i|pos)}{P(w_i|neg)}\right) \Rightarrow \log\frac{P(pos)}{P(neg)} + \sum_{i=1}^{n}\log\frac{P(w_i|pos)}{P(w_i|neg)}$$

Although this explanation is based on binary classification, it is not hard to implement Naïve Bayesian classifier to multiclass classification

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

Where denominator $\mathbf{P(x_1, \ldots, x_n)}$ can be omitted cause its always the same and base decision hypothesis has maximum probability, that is known as Maximum A Posteriori (MAP) where y class is predicted label C_k out of k labels as:

$$\hat{y} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}}\; p(C_k)\prod_{i=1}^{n} p(x_i \mid C_k)$$

Apparent from its naïve assumption of independency of the features, this algorithm sometimes performs very well in document classification and spam recognition problems [36]. Its major advantage is that it is computationally not expensive and can be extremely fast in comparison with more sophisticated algorithms. It is used for very large data sets and on the other hand doesn't require a lot of training data.

4.2 Decision Trees

Decision tree algorithm, starting from the root node it makes decision on which attribute to split with greedy approach that measures consequences of every possible attribute-based partition. The goal at every partition is to split the data in the most homogenous segments possible. Every attribute-based partition is measured by entropy after partition and decision is based on choosing the most information gain or difference before and after partition.

This process can take a lot of computational time, especially when data has a lot of features, which make training process longer than for some other algorithms. Also, in a case attribute values are continuous, it's necessary to transform them into discrete values (binding). Attributes with large number of values are favored over attributes with small number of values. They become root attributes and make broad tree that performs poorly on unseen instances. It can be overcome with alternative method for measuring information gain (gain ratio) that penalize attributes with large numbers of values.

It's prone to overfitting, with very complex tree structure, that doesn't perform well on a test data. In this situation, stopping the further growth or pruning the tree, setting the maximum number of the leaf nodes (final nodes), or maximum depth of the tree can help [37]. Test on validation set is one of a way how to decide when to stop tree partitioning. Of course, all of this make computation more complex.

On the other hand, it doesn't require a lot of data preparation. Decision tree is reliable comprehensive model with considering all possible decisions and making analysis. It can be represented in easier way to understand in some cases, set of if-then-else decision rules. But in classification it can have very complex structure cause number of features is extremely high.

4.3 K-Nearest Neighbors

The k-nearest neighbors (KNN) algorithm can be used both to solve classification and regression problems. There is no real process of training the model. Transforming data attributes in feature vectors and saving it with corresponding labels is all that is done in this phase. Computation starts when it comes to evaluation of new unlabeled cases. It makes it very slow in evaluation and considered as one of the lazy learning algorithms. On the other hand, training process is much easier and faster than in more complex algorithms.

The main idea is that data points who are close to each other in vector space should have same classification labels. So, estimation of new unlabeled sample is based on maximum between neighbor's labels for specified number of neighbors k. This number k is chosen by user, usually through running multiple cross-validation tests.

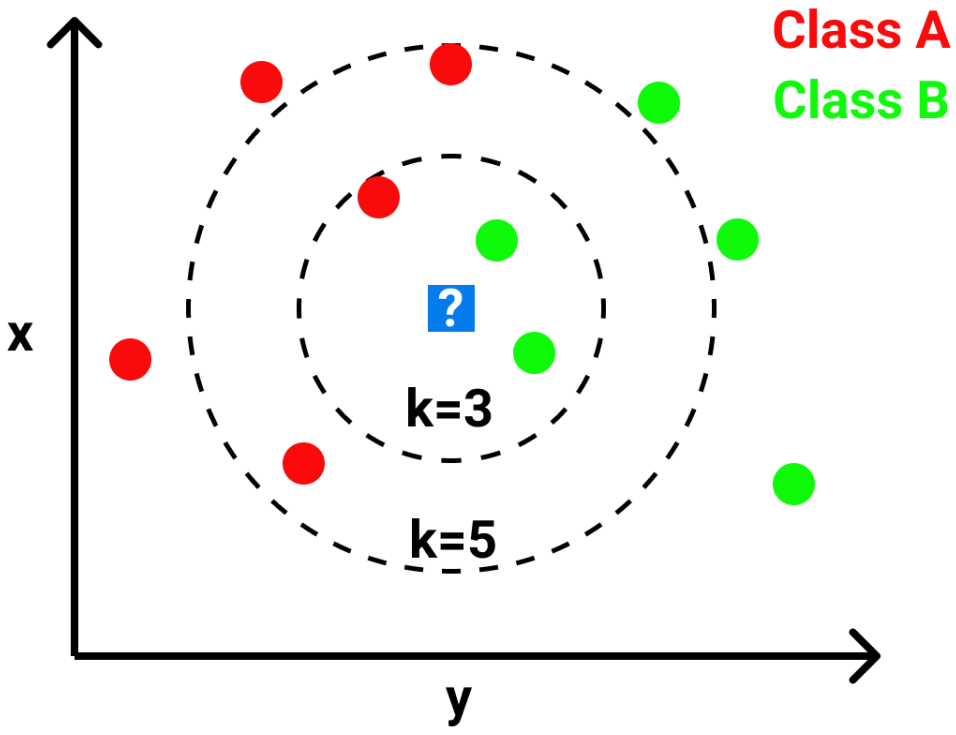

Figure 2: A simple representation on how k affects decision

The number of neighbors is not only a parameter that affects accuracy of algorithm. Considering the fact that the most common way of measuring vector's distance is the Euclidean distance:

$$D(a,b) = \sqrt{\sum_{i=1}^{n}(b_i - a_i)^2}$$

It can be very wrong if the training data set is unbalanced, with some class having majority upon the others. Contribution of the neighbor is usually divided with its distance to query point in order to get better performance. One of the main disadvantages of the KNN is that decision is based on distance of all attributes from the query point. It is quite opposite from decision tree, based on measuring contribution of an attribute in categorization and it can lead to prediction based on unimportant features. This can be very serious issue, when there are many irrelevant attributes in the data set, as it can be the case in text classification tasks. Weighting importance of attributes or reducing features can improve performance in these situations [38].

4.4 Support Vector Machine (SVM)

Support Vector Machine (SVM), can be a very good choice for linearly non separable data. It maps data features to high dimensional vector space by adding features to it, but in much more computationally efficient way than adding high order polynomials in logistic regression. Functions that are used in this process are kernel functions.

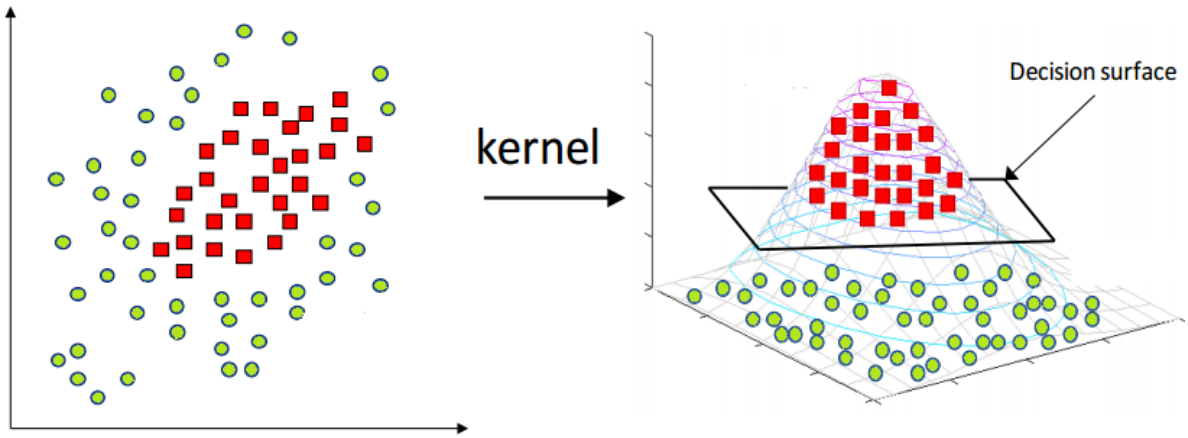

Figure 3: Kernel (figure by Grace Zhang) [39]

The figure above shows how is two-dimensional linearly inseparable data transformed to three-dimensional data with plane as decision surface. In machine learning problems, data is multidimensional and after transforming it into higher dimensionality, computations are much more complex and expensive. Using kernel method reduces this computation to originally dimensionality and that is called “kernel-trick". “The kernel corresponds to a dot product in a

(usually high-dimensional) feature space. In this space, our estimation methods are linear, but as long as we can formulate everything in terms of kernel evaluations, we never explicitly have to compute in the high-dimensional feature space" [40].

There are different kernel types, such as linear, polynomial, Gaussian, Radial Basis Function, or RBF, and sigmoid. Kernel is similarity function that provide more complex features from original data.

SVM can be used without kernels, when data set is large and number of features is small. The goal of SVM in general is to find the largest separation or margin between the classes, thus SVM is also known as large margin algorithm. Support vector points are points closest to the margin or a hyperplane and only this subset of data points is used in the predictions for new samples, that improves memory efficiency. One more advantage of SVM is that is very accurate in high-dimensional spaces.

On the other hand, SVMs do not make probability estimates, which are desirable in some classification problems. They are not very efficient computationally if dataset is very big. But SVM is very effective in text mining tasks, particularly due to its effectiveness in dealing with high-dimensional data. It is used for detecting spam, text category assignment and sentiment analysis.

Chapter 5
Web Text Classifier (Katigoriopoiitis)

The previous chapters approached the problem of utilizing web texts. Several tools and algorithms that can be used for web classification were presented. It was decided that a real-time tool that offering the benefits of a Web search engine in combination with text classification would contribute as an addition in the existing toolset. As key aspects of classification considered the extraction of: language, title, keywords, author, created/updated timestamps, the main text content, the content type and multimedia links related to the content. It was decided that the tool must be able to be trained for multiple domains and purposes. Based on the extracted content, the classifier must be able to make predictions by using one or many selected models.

Third party systems can use Classifier as a backend. Classifier can perform for example as a backend system for an advanced search engine. Another use case of classifier is to perform as a crawler's middleware. So, crawler can send retrieve and store the content classified.

5.1 Specifications

1. The tool is able to classify a web page based on classification models.

2. Classification models can be created by posting labeled training sets in CSV, Excel or JSON format to the API. The user is able to define n-grams, stemming and stop words preferences and the algorithm of the model. The system responds with a model id that can be used in classification requests.

3. The tool is able to extract timestamps, keywords, authors and the text language LD of each page, based on the meta tags, Open Graph meta tags and JSON-LD.

4. The tool is able to detect the language of the content based on meta tags, as well as on content analysis.

5. The tool is able to get the main content of the page out of the boilerplate and any html tags.

6. The tool can have as input one or multiple URLs and will respond with the classified data.

7. The tool can perform a web search based on search terms and respond with the classified data for all the results. The search features offer pagination for more results as well as number of results per page.

8. The search is able to be limited in a list of specified websites.

9. The tool can take as parameter one or many trained model ids for classification.

10. The tool is able to train models in the background and offers a way to check the training status.

11. The tool calculates model accuracy by splitting the training set in 70% training and 30% testing.

5.2 Content Extraction

HTML templates can be complex to analyze. There is a variety of implementations but there is no standard method of separating the content of the side information[14]. The tool needs to be able to extract the content. Two algorithms were considered as options.

Readability[15] achieved some positive results of extracting the main text of Web articles [41]. Readability algorithm is very simple, based only in structural features of the HTML DOM elements. It is mainly powered by a guess that DOM elements are labeled with common class names that describe their kind such as **content**, **article**, **hidden**, **banner** etc. There is a collection of positive and negative class names. When a DOM element is marked as negative, it is excluded of the output, when it is ranked as positive, it is considered as content. Neutral elements without any positive or negative classes are depending on parent elements. The algorithm takes into consideration additional information such as heading tags but the main issue is that it does not take into account any text features.

Boilerpipe[16] is another text extraction algorithm. In a paper published by Christian Kohlschütter, Peter Fankhauser and Wolfgang Nejdl in 2010, another approach was proposed in order to achieve more accurate results in Web Data extraction. The paper enumerates the features that can be used in order to separate the main content and the boilerplate[17] of a Web page. In order to achieve a *web scalable domain-independent solution*, the paper is focusing on shallow text features that are language independent. Quantitative features such as *average word length* and *average sentence length* can be used. Additionally, in order to avoid "overfitting to a particular data set or domain" and keep the solution generic, the authors decided to focus only on limited structural features such as: "The presence of a particular headline tag (H1, H2, H3, H4, H5, H6), a paragraph tag (P), a division tag (DIV) and the anchor text tag (A) as an HTML element that encloses a particular text block". Unlike the Readability's algorithm approach, the class names of the elements are not in consideration because "the more CSS is used, the less important the semantics

[14] A recently published book of Jay M. Patel analyzes the problem of web content extraction in depth [69].
[15] The project as mentioned in previous chapter is not anymore active, however plenty of forks and adaptations of the original implementations can be found. For the needs of this dissertation, a python implementation of the algorithm was used [67].
[16] For the needs of the classification tool, a Python port of Boilerpipe was tested and used [68].
[17] As boilerplate considered the information that is not related with the main content of a Web Document.

of an HTML tag becomes […] unfortunately, CSS classes and sequences of HTML tags are inherently site- and document-specific". It is important to mention that rendering the entire page including CSS styles considered as a potential approach that would give a cleaner picture of how the content blocks are related to each other, however, this is a "computational expensive operation" that is not scalable and further, training a model in specific templates, considered as non-reusable solution. The paper shows "that a combination of just two features - **number of words** and **link density** - leads to a simple classification model that achieves competitive accuracy" [42].

Boilerpipe algorithm is far from perfect. However, it achieved good results in news web pages as well as in blogs. So, it was chosen as the main method of content extraction. Boilerpipe fails to give good results in web pages lacking main content as for example social media pages or main website pages. If no content can be extracted, the metadata content is used for classification.

5.3 Text Classification Techniques

Text classifier needs to implement techniques of Natural Language Processing[18]. There are many different techniques to automate text classification. For example, a simple keyword search in texts in order to specify the category of the text can achieve some results. However, more advanced machine learning techniques are needed in order to achieve high accuracy in text classification. Katigoriopoiitis incorporates all the following techniques in its classification process.

5.3.1. Pre-processing methods

"Text pre-processing is an essential part of any NLP system, since the characters, words, and sentences identified at this stage are the fundamental units passed to all further processing stages, from analysis and tagging components, such as morphological analyzers and part-of-speech taggers, through applications, such as information retrieval and machine translation systems" [43].

Usually text contains characters or symbols that are not important for the classification, resulting in adding unwanted noise in the dataset. Removing symbols, whitespace, or even numbers is a

[18] "Natural language processing (NLP) is an integral area of computer science in which machine learning and computational linguistics are broadly used" [62].

common pre-processing technique. Additionally, eliminating contractions in text can be beneficial in order to remove duplicate features. For example, **it's** and **isn't** will be converted to **it is** and **is not** respectively. This way the meaning of the tokens can be unified.

Another pre-processing method is the stop words elimination. As stop words are considered the very common words such as pronouns and prepositions. This kind of words can potentially appear in all contexts or types of texts so they are not important in distinguishing classes. Stop words list can be different based on the kind of the text. For example, in scientific articles classification the list of stop words is including not only very common words, articles, pronouns and prepositions, but additional common scientific words. In some cases, stop words elimination is not a desired technique, however, it is heavily used in text classification preprocessing.

Other common pre-processing methods can be the punctuation elimination and the character case elimination. In case of character case, there are techniques to keep names and abbreviations untouched. In order to avoid for example, the **US** (United States) abbreviation to be mixed with the word **us**. A simple technique is to leave untouched words with capital letters in the sentence and eliminate only capital letters of the first word of each sentence.

5.3.2. Tokenization

"Concepts such as word, collocation, and multimorphemic item are important to lexicographers and linguists; whereas the concept of token is specific to certain processes in NLP and MT. A token will not be broken down into smaller parts, lit other words, for the purpose of computational processing, it can be treated as an atom" [44].

While Machine Learning algorithms can achieve amazing accuracy in text classification, in reality the mechanism under the hood is far different of the human brain. Machines are able to understand text only as chunks of characters converted to numbers. The main idea of text classification lays on frequencies of chunks called tokens. Based on the goal, a token can be character based, syllable based, word based or even sentence based. One of the most common techniques in text classification, the Bag-of-Words (BoW), is based in word tokens. The phase of splitting the text in tokens is called tokenization [45].

5.3.3. Word Normalization

Since computers are not able to know and apply grammar rules, words with the same meaning need to be normalized in a basic form, ideally into lemmas. This process is called Lemmatization and it (ideally) demands a thesaurus dictionary available in order to be accurate. In lemmatization for example, the words **cars**, **car's**, **care**, **caring**, **walking**, **visually** and **visuals** will be converted to **car, car**, **care**, **care**, **walk**, **visual** and **visual** respectively. As it is obvious in the example, since visually and visuals sharing the same lemma (and so most probably meaning), will be treated as the same word same for **care** and **caring** as well as **cars** and **car's**. In 2018, Toms Bergmanis and Sharon Goldwater introduced a lemmatization model based on neural networks. The model was trained in 20 languages and proved to perform well with low resource training sets [46].

Often, it is not easy or even not necessary to get lemmatization done. Another process that can achieve similar results called Stemming can be applied. Stemming, based on very basic grammar patterns, erases suffixes of words in order to achieve something close to lemma. For example, in English language a very simple stemmer could remove suffixes like **ing**, **ed, s, 's, ive, ively, able, atible, fy** and **ly**. Considering the same example of **cars**, **car's**, **care**, **caring**, **walking**, **visually** and **visuals**, by applying the above rules the outcome is **car, car, care, car, walk, visual** and **visual** respectively. While it can achieve some results, stemming has two main weaknesses. It can lead to wrong lemmas like in case of **caring** and **car**. Also, it can lead to none existing words (for example **stepping** would become **stepp**), since the stemmers are following very basic grammar rules. Good stemmers usually have some additional functionalities, such as a dictionary of irregular verb forms. For example, **are, is, am** would be converted to the same word **be**. One of the very first stemming algorithms for English language created by Porter in 1980. Porter's algorithm, managed to reduce a vocabulary of 10000 words down to 6370 distinct entries [47].

Lemmatization and Stemming are limiting the number of features in order to avoid overfitting of the model. However, it is important to mention that eliminating important grammar information such as tenses, is not desirable in some cases. Word normalization can take place either before or after tokenization, depends on the implementation. Despite the fact lemmatization gives slightly better results, due to better performance [48], classifier is using Stemming and not Lemmatization since Stemming is simpler and so, faster.

5.3.4. Feature extraction and NLP Models

When text pre-processing tokenization and normalization is done, tokens need to be transformed into features. This phase is called feature extraction and like the preview's steps, it can be done in different ways, depending on the chosen technique and based on the nature of the data. Translation of a human language[19] to machine language[20], words with different semantic meanings based on context, distinguishing important terms for analyses from a bunch of others, understanding linguistic norms and grammar rules, all of these are necessity in complex tasks such as auto complete text recommendations.

During feature extraction, tokens can be treated as independent or paired in continues sequences called N-grams[21]. N-gram models, revealing patterns in sequences of words. For example, in the simple Bag-of-Words approach, the words "Corona" and "Virus" would be treated separately, while their relationship is important. By using a bigram model, "Corona Virus" will be appeared as a feature. A trigram model for example could reveal "Corona Virus pandemic". It is obvious that very big N-grams can lead to overfitting so they must be selected carefully. Additionally, combination of N-grams proven to be effective. So, for example features can be extracted from both unigrams and bigrams in the same model.

After choosing the vocabulary of terms that are the best for representation of a document, it is necessary to transform that vocabulary to an algebraic model as an input for machine learning algorithms, also known as vector-space model. This is maybe the most important task of Data Mining. There are different formats of this model and the choice is based on the purpose of the analysis. The document (or training sample), is represented as a vector with dimension equal to the number of terms in the data set vocabulary. It has non-zero values if that specific term exists in the document (training sample).

The simplest and oldest model is the Standard Boolean Model of Information Retrieval [49]. A vector of zeros and ones is saving binary information that represents whether the term (feature) exists. This model does not count frequencies. Further research led to more sophisticated models, that are giving weight to a term appearance.

[19] Natural Language
[20] Vectors
[21] N-grams are called differently based on the value of N: 1-grams are called unigrams, 2-grams are called bigrams, 3-grams are called trigrams etcetera.

Term frequency model (TF) is a vector that is saving information of a frequency of a term in a document. The issue with TF is that in some cases terms are more frequent, but not important for distinguishing the meaning of a document, as they appear a lot in all documents of a data set.

Term Frequency-Inverse Document Frequency (TF-IDF) model is made to penalize term frequency by dividing with frequency of a term in all documents. In text classification, TF-IDF doesn't necessarily perform well, because it penalizes also terms important for specific class. It led to further researches in this field, like Supervised term Weight (STW), that is using training based on class for making input for training model. "Supervised term indexing leverages on the training data by weighting a term according to how different its distribution is in the positive and negative training examples" [50].

In the tasks such as Sentiment Analysis, Neural Machine Translation and Question Answering, that demand deeper understanding of the context, an advanced model named Word Embedding shows very good results. The idea is based on the concept that words with similar meaning should have similar representation, that make this representation capable to save semantic meaning of a word. Words are represented as a vector of features that are different aspects of semantic meanings and trained as a model. However, for simple text classification, some more advanced techniques can be avoided in order to make computation simpler. As can be seen in the **Appendix A**, TF-IDF with combination of N-Grams can give very good prediction accuracy when SVM is used. A unigram, bigram and trigram combination decided to be used for feature extraction.

5.3.5. Dataset

Beside the techniques of converting natural language texts to features and vectors, the most important part of machine learning is the dataset. The dataset needs to be balanced and representative. A balanced dataset is a dataset that represents all the categories equally. If for example a model aims to classify text into "political articles" and "tech articles", the amount of texts of each category must be equal.

A common practice in model training is to split the dataset into training dataset and test dataset. The reason is simple: while model training is done, prediction accuracy has to be measured. If the training dataset used for measuring the accuracy the results will be unrealistic, while the test dataset will contain unknown data and will give a better accuracy estimation. A possible split is for example to use 70% for training and 30% for testing. This split portion was used in the classifier.

5.4 Ontologies and Semantic Web

Ontologies caring semantic information about the meanings and the relationships of terms. “From an object-oriented point of view, ontologies are domain classes that contain logical statements that make their meaning explicit. […] tools called reasoners can exploit these logical statements to perform advanced queries which reveal implicit relationships among resources” [51]. Semantic Web created in order to cover the weaknesses of HTML in incorporating semantics markup. “People can read and understand a web page easily, but machines can not. To make web pages understandable by machines, additional semantic information needs to be attached or embedded to the existing web data.” [52]. The representation of semantics had several different standards during the years. One of the most popular is the RDF [53] framework. There is an XML syntax called RDF XML as well as a JSON version called RDF JSON. Web ontologies and Metadata have important impact on search engine optimization (SEO) and should be taken in consideration for Web text classification. Search engines cooperated in order to force a standard object schema for web semantics [54]. Social networks on their side, forced other standards. Facebook proposed the Open Graph protocol which was based on RDF [55]. Twitter is using its own meta tags prefixed by “twitter:”, in combination with Open Graph meta tags [56]. Furthermore, JSON-LD [57] is one of the standards that can be found in plenty of modern websites.

The classifier was designed to detect basic information such as author, publishing date, updated date and content description [22] based in meta tags, JSON-LD and Open Graph. A **MetadataWrapper** class was created in order to merge both information sources in a universal structure. For JSON-LD, a custom parser was created. Meta tags and Open Graph meta tags are extracted by using the metadata_parser python library [58]. The code can be found in the **Appendix C**.

5.5 Web Search

Web search is powered by Microsoft Azure Web Search API. The Search API returns the results list with titles, URLs and a very small text description. Classifier retrieves the URL list and fetches the content for each one. In order to improve the performance, the multithreading library is used

[22] Additionally, the tool tries to detect keywords, news_keywords and multimedia related to the content.

and each URL is fetched, parsed and classified in a different Thread. When all threads are done, the result set returns as a response.

5.6 Limitations

The tool needs to fetch each web page separately. This can cause delays in responses. If the tool is planned to be used from web applications, timeouts can be caused if big number of URLs or Web search results per page is requested. In this case, is recommended to limit the web results up to 10 per page. Additionally, since the extraction of the data is based on Boilerpipe algorithm, Katigoriopoiitis fails to extract the main content in Web pages lacking of main text body. For example, it would perform pretty well in a newspaper article where the main text is much bigger than any other part of the page and poorly in a social media page, where the main content is spread in many posts.

Chapter 6
Conclusion

An important theoretical and practical work has been already done in the field of data mining. While the internet data is growing, more efficient tools are needed. This forces Internet technologies and standards to get richer and this maximizes the options for data mining. On the other hand, Internet data as well as the used base are growing exponentially adding additional obstacles and concerns. As presented in the dissertation, content classification can be easily achieved by using simple machine learning model implementations. However, more sophisticated models are needed in order to achieve high accuracy. This dissertation focused on supervised machine learning. The contribution of the dissertation lays on the deliverable system (Katigoriopoiitis). The ways of efficient classification of web data, as presented in the previous chapters, lay on multiple techniques: Automated web scrapping, data extraction strategies and machine learning algorithms are needed in order to achieve good results. All this work was integrated in the classifier and presented in this dissertation. The deliverable tool tried to give a generic and flexible solution in web classification. The ability of creating multiple models makes classifier efficient enough to cover both genre and content classification.

Chapter 7
Future

Katigoriopoiitis was built as a web API service. As a future research, the design and implementation of a web and a GUI application can be created. The tool detects videos and images related to the content. Image recognition models could extend the usability of the tool. The tool needs to fetch the full content of each URL in order to extract the data. Breaking the process in multiple threads reduced a lot the processing time; however, a more scalable architecture could be designed. Additionally, the implementation of web crawling functionality can expand the usability of the tool by creating labeled corpora in real-time. The datasets used in this dissertation were limited in size and variation. In the future Katigoriopoiitis could be improved by adding more and richer pre-trained models. Finally, further research in Web text extraction algorithms would aim to improve the performance of Katigoriopoiitis in social media pages.

Bibliography

[1] S. Bowerman, Norms and Correctness. In Brown, K. (ed.): Encyclopedia of Language and Linguistics, Second Edition, vol. 8, Oxford: Elsevier, 2006, p. 701–703.

[2] B. Kessler, G. Nunberg and H. Schutze, Automatic Detection of Text Genre, Xerox Palo Alto Research Center, Department of Linguistics, 1997.

[3] S. N. Alsubari, M. B. Shelke and S. N. Deshmukh, Fake Reviews Identification Based on Deep Computational Linguistic Features, vol. 29(8s), International Journal of Advanced Science and Technology, 2020, pp. 3846-3856.

[4] V. R. Benjamins, J. Contreras, O. Corcho and A. Gomez-Perez, Six challenges for the Semantic Web, 2002.

[5] M. Najork, Web Crawler Architecture, Microsoft Research, 2009.

[6] "Top free software for text analysis, text mining, text analytics," [Online]. Available: https://www.predictiveanalyticstoday.com/top-free-software-for-text-analysis-text-mining-text-analytics/. [Accessed 2020].

[7] "Corpora: Useful web tools," [Online]. Available: http://www.englicious.org/lesson/corpora-and-web-tools/useful-web-tools. [Accessed 2020].

[8] "Natural Language Toolkit," NLTK Project, [Online]. Available: https://www.nltk.org. [Accessed 2020].

[9] A. Loupasakis, "Greek Stemmer," [Online]. Available: https://pypi.org/project/greek-stemmer/. [Accessed 2020].

[10] "Git," [Online]. Available: https://git-scm.com. [Accessed 2020].

[11] "GitHub," [Online]. Available: https://github.com. [Accessed 2020].

[12] "Heroku," SalesForce, [Online]. Available: https://www.heroku.com. [Accessed 2020].

[13] "Postman," Postman, Inc, [Online]. Available: https://www.postman.com. [Accessed 2020].

[14] "Figma," [Online]. Available: https://www.figma.com. [Accessed 2020].

[15] D. Greene and P. Cunningham, Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering, Proc. ICML., 2006.

[16] J.-N. Mazón, J. J. Zubcoff, I. Garrigós, R. Rodríguez Ortega and R. Espinosa Oliva, Open

Business Intelligence: On the importance of data quality awareness in user-friendly data mining, Conference: Proceedings of the 2012 Joint EDBT/ICDT Workshops, 2012.

[17] We Are Social & Hootsuite, Digital 2019 Global Digital Overview, retrieved from https://datareportal.com/reports/digital-2019-global-digital-overview, 2019.

[18] D. J. Hand and N. M. Adams, Data Mining, Wiley Online Library, 2015.

[19] T. M. Michell, Machine Learning, McGraw-Hill Science, 1997, p. 1.

[20] A. L. Samuel, Some Studies in Machine Learning Using the Game of Checkers, IBM J. Res. Dev. 3, 1959, pp. 210-229.

[21] T. M. Mitchell, Machine Learning, McGraw-Hill Science, 1997, p. 2.

[22] M. Cord and P. Cunningham, Cognitive Technologies, Springer-Verlag, 2008, pp. 21-46.

[23] M. Cord and P. Cunningham, Cognitive Technologies, Springer-Verlag, 2008, pp. 51-87.

[24] The Apache Software Foundation, OpenNLP, retreaved from http://opennlp.apache.org/docs/1.9.3/manual/opennlp.html, 2020.

[25] Orange Python, retrieved from https://orange.biolab.si/download/, 2020.

[26] Orange Data Mining, retrieved from https://orange.biolab.si, 2020.

[27] V. Vryniotis, DatumBox Machine Learning API, retrieved from http://www.datumbox.com, 2020.

[28] Google, Cloud Natural Language API, retrieved from https://cloud.google.com/natural-language/docs/quickstarts, 2020.

[29] The Guardian Open Platform, retrieved from https://open-platform.theguardian.com, 2020.

[30] WebCorp, retrieved from http://www.webcorp.org.uk/live/guide.jsp, Research and Development Unit for English Studies (RDUES), 2020.

[31] A. Renouf, A. Kehoe and J. Banerjee, WebCorp: An integrated system for web text search. Language and Computers, 2006, pp. 47-67.

[32] BootCat, retrieved from http://bootcat.dipintra.it, DIT/DipInTra (ex SSLMIT), 2020.

[33] SketchEngine retrieved from https://www.sketchengine.eu, Lexical Computing, 2020.

[34] Instaper, retrieved from https://www.instapaper.com, 2020.

[35] "Digital Research Tools (DiRT)," [Online]. Available: https://digitalresearchtools.pbworks.com/w/page/17801682/Linguistic%20Tools. [Accessed 2020].

[36] T. M. Mitchell, Machine Learning, McGraw-Hill, 1997, p. 181.

[37] T. M. Mitchell, Machine Learning, McGraw-Hill, 1997, pp. 68-69.

[38] T. M. Mitchell, Machine Learning, McGraw-Hill, 1997, p. 235.

[39] G. Zhang, What is the kernel trick? Why is it important?, medium.com, 2018.

[40] T. Hofmann, B. Schölkopf and A. J. Smola, Kernel methods in machine learning, vol. 36, Annals of Statistics, 2008.

[41] D. Frakes, Readability makes Web pages more readable, MacWorld, 2009.

[42] C. Kohlschütter, P. Fankhauser and W. Nejdl, Boilerplate Detection using Shallow Text Features, L3S Research Center / Leibniz Universität Hannover, 2010.

[43] S. Kannan and V. Gurusamy, Preprocessing Techniques for Text Mining, Madurai Kamaraj University.

[44] J. J. Webster and C. Kit, Tokenization as the initial phase in NLP, 1992.

[45] P. Cunningham, M. Cord and S. J. Delany, Study of Various Methods for Tokenization, Supervised Learning. In: Cord M., Cunningham P. (eds) Machine Learning Techniques for Multimedia. Cognitive Technologies.: Springer, 2008.

[46] T. Bergmanis and S. Goldwater, Context Sensitive Neural Lemmatization with Lematus in Proceedings of NAACL-HLT, Association for Computational Linguistics, 2018, p. 1391–1400.

[47] M. F. Porter, An algorithm for suffix stripping, 1980.

[48] A. Kutuzov and E. Kuzmenko, To lemmatize or not to lemmatize: how word normalisation affects ELMo performance in word sense disambiguation, 2019.

[49] F. W. Lancaster and E. G. Fayen, Information Retrieval On-Line, Melville Publishing Co, 1973.

[50] F. Debole and F. Sebastiani, Supervised Term Weighting for Automated Text Categorization, vol. 138, In: Sirmakessis S. (eds) Text Mining and its Applications. Studies in Fuzziness and Soft Computing, 2004, pp. 81-97.

[51] H. Knublauch, D. Oberle, P. Tetlow and E. Wallace, A Semantic Web Primer for Object-Oriented Software Developers, W3C, 2006.

[52] Z. Ding, Y. Peng and R. Pan, BayesOWL: Uncertainty Modeling in Semantic Web Ontologies. In: Ma Z. (eds) Soft Computing in Ontologies and Semantic Web. Studies in Fuzziness and Soft Computing,, vol. 204, Springer, 2006, pp. 4-5.

[53] "Resource Description Framework (RDF)," W3C, 2014. [Online]. Available: https://www.w3.org/RDF. [Accessed 2020].

[54] "JUNE 2 2011 Introducing Schema.org: Bing, Google and Yahoo Unite to Build the Web of Objects," 2011. [Online]. Available: https://blogs.bing.com/search/2011/06/02/introducing-schema-org-bing-google-and-yahoo-unite-to-build-the-web-of-objects/. [Accessed 2020].

[55] "The Open Graph protocol," Facebook, [Online]. Available: https://ogp.me/. [Accessed 2020].

[56] "Twitter Cards," Twitter, [Online]. Available: https://developer.twitter.com/en/docs/twitter-for-websites/cards/guides/getting-started. [Accessed 2020].

[57] "JSON for Linking Data," [Online]. Available: https://json-ld.org. [Accessed 2020].

[58] J. Vanasco, "MetadataParser Pyhton Library," [Online]. Available: https://pypi.org/project/metadata-parser/. [Accessed 2020].

[59] T. Berners-Lee, Information Management: A Proposal, CERN, 1989.

[60] T. Berners-Lee and R. Cailliau, WorldWideWeb: Proposal for a HyperText Project, CERN, 1990.

[61] V. G. Cerf and R. E. Kahn, A Protocol for Packet Network Intercommunication, IEEE, 1974.

[62] A. Jain, G. Kulkarni and V. Shah, Natural Language Processing, vol. 6, International Journal of Computer Sciences and Engineering, 2018, pp. 161-167.

[63] S. K. Singh, V. Prasad, S. Tanwar and S. Tyagi, Software as a Service, 2019.

[64] Arc90, Readability Tool Fork, retrieved from https://ejucovy.github.io/readability, 2020.

[65] M. Alrizah, S. Zhu, X. Xing and G. Wang, Errors, Misunderstandings, and Attacks: Analyzing the Crowdsourcing Process of Ad-blocking Systems, in IMC '19: Proceedings of the Internet Measurement Conference, Association for Computing Machinery, 2019.

[66] J. Pomikalek, Removing Boilerplate and Duplicate Content from Web Corpora, Masaryk University, Faculty of Informatics, 2011.

[67] "Readability Algorithm Python Port," [Online]. Available: https://github.com/phensley/python-readable/blob/master/readable/core.py. [Accessed 2020].

[68] J. Riebold, "Boilerpipe Python Port (boilerpy)," [Online]. Available: https://pypi.org/project/boilerpy3/. [Accessed 2020].

[69] J. M. Patel, Getting Structured Data from the Internet, Apress, 2020.

Appendix A
Model Accuracy Comparison

Accuracies of different text classification models were tested on the same dataset. The dataset contained about 2225 documents from the BBC news of Politics, Entertainment, Technology, Business and Sports. The 70% (1557 documents) of the dataset were used for training and the other 30% (668 documents) for testing the accuracy in unknown data. The experimental models were based on the algorithms: **SVM**, **Decision Tree**, **K-Nearest Neighbors**, **Bernoulli Naïve Bayes** and experimentally, **Gaussian Naïve Bayes** which is usually used for regression problems rather than classification.

All five algorithms were tested in four different N-Gram combinations giving a total of twenty different classification models:

- Unigram
- Unigram & Bigram
- Unigram & Bigram & Trigram
- Bigram

In all models (except Bernoulli Naïve Bayes implementation which has a binary representation) TF-IDF model was used for feature representation. The experiment was implemented in Python by using the **sklearn**, **nltk** and **pandas** libraries. The results are presented in **Figures 6 - 9**. The main conclusion is that the SVM algorithm performed well in all N-Gram combinations while Bernoulli Naïve Bayes accuracy dropped significantly in some models.

Adding bigrams and trigrams improved language correlations, but on the other hand added more features to the data. In **Figure 4**, we can see how the vocabulary of the training dataset grows

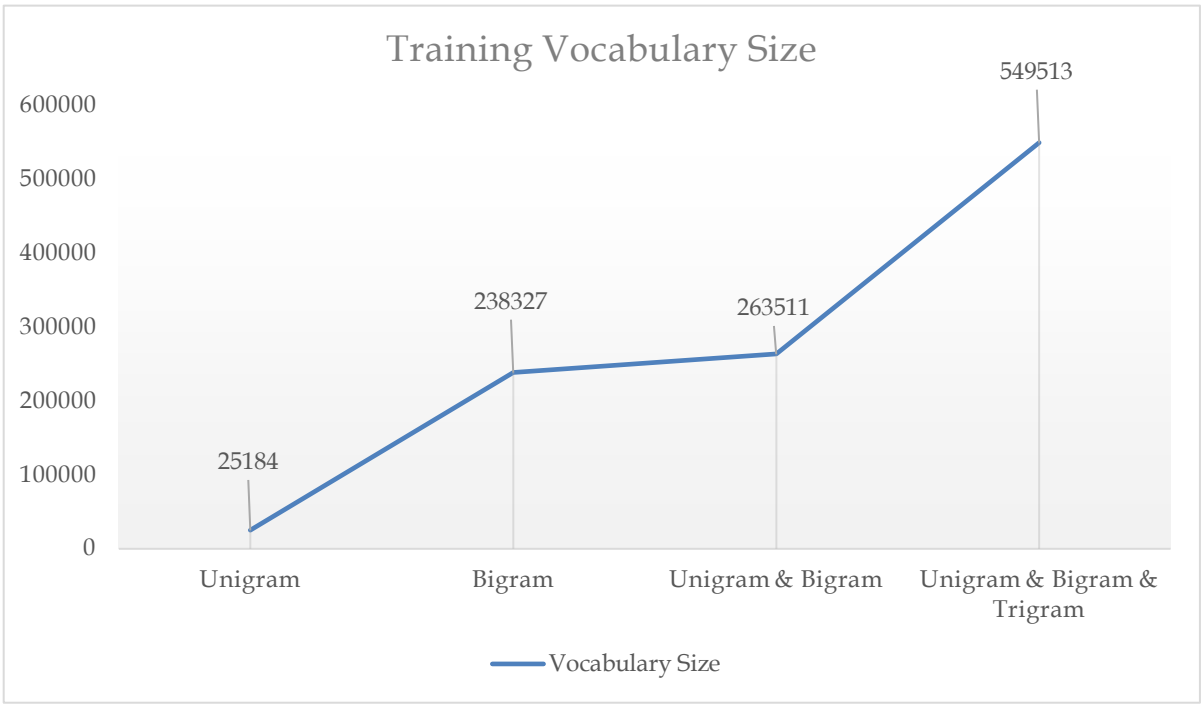

Figure 4: Training Dataset Vocabulary

(and as a result, the features). This led to models with sparse data. The values of these n-gram features that were added to vocabulary will be represented mostly as zeros. In Bernoulli decision rule, these zero values are penalized and that is the reason of dramatical dropping of the accuracy. We can notice also in Decision Tree slightly dropping of the accuracy in the models with more features. On the other hand, in K-Nearest Neighbor models the accuracy is slightly increasing by adding more features. Combination of n-grams showed the best results in SVM models. The accuracy is improving by taking more features in consideration; as it could be expected because, as was mentioned in fourth chapter of this dissertation, SVM model performs well in high dimensional vector space.

Finally, using only bigram features is not showing good results. Logically, unigrams (or single words) have more power in differentiation of documents, while just some bigrams are important. For example, from the Shakespeare's phrase: "**Jealousy is the green-eyed monster**", we would get the bigrams: "**Jealousy is**", "**is the**", "**the green-eyed**" and "**green-eyed monster**". The bigram "**green-eyed monster**" seems to have some sense considering document classification while the others could add unwanted noise in the training data. It could be concluded that it would be good in the future to try reduction of features with just certain n-grams. In this study, Gaussian Naïve Bayes was experimentally tested on discrete data and it is the only model in this test that improves with in the bigram model. Further research is needed in order to find the reasons of this behavior.

Another aspect of poor accuracy in some models lays on the nature of the dataset. In this experiment the dataset was relatively small; this fact enhances can lead to overfitting and so, in poor performance in models that bigrams or trigrams are used. Some models seemed to be very sensitive in overfitting. Additionally, the dataset was taken only from BBC news and it is expected to have less accuracy on tests from real world data. As can be seen in **Figure 5**, the test dataset in unigram models found to have only 22.86% unknown vocabulary (that was not included in the training dataset) while in models containing bigrams and trigrams the great majority of the testing set vocabulary is unknown.

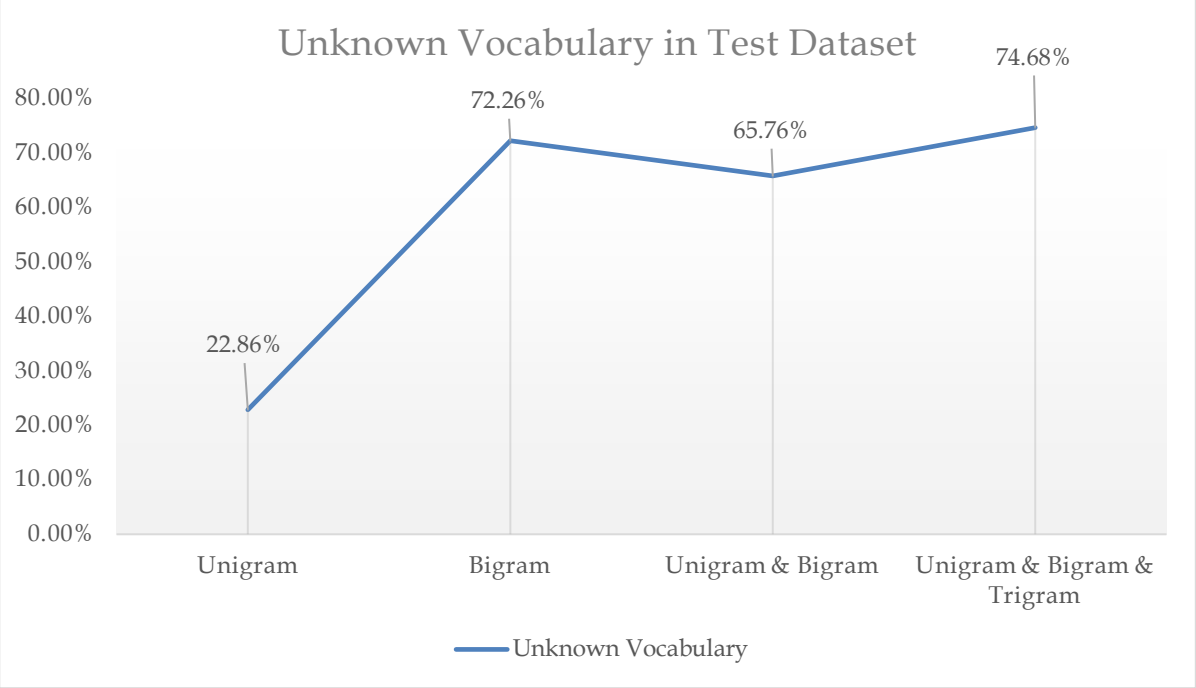

Figure 5: Unknown vocabulary in test dataset

The study was done on simple document classification and it showed some improvement with adding of features. It is assumed that this improvement would be higher in a task of sentiment analysis, where for example missing negation before word could really lead to misclassification. The conclusion is that adding of features should be carefully applied based on task and algorithm that is best for data set. It is not always improving accuracy, but in this relatively small data set, it showed very good results in combination with SVM classifier.

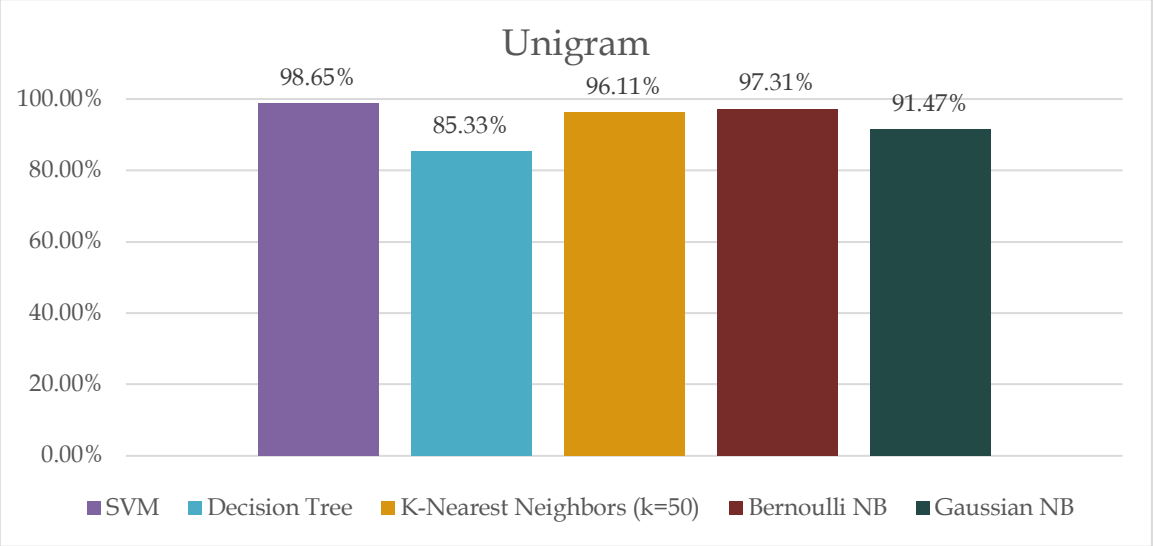

Figure 6: "Unigram" model performances

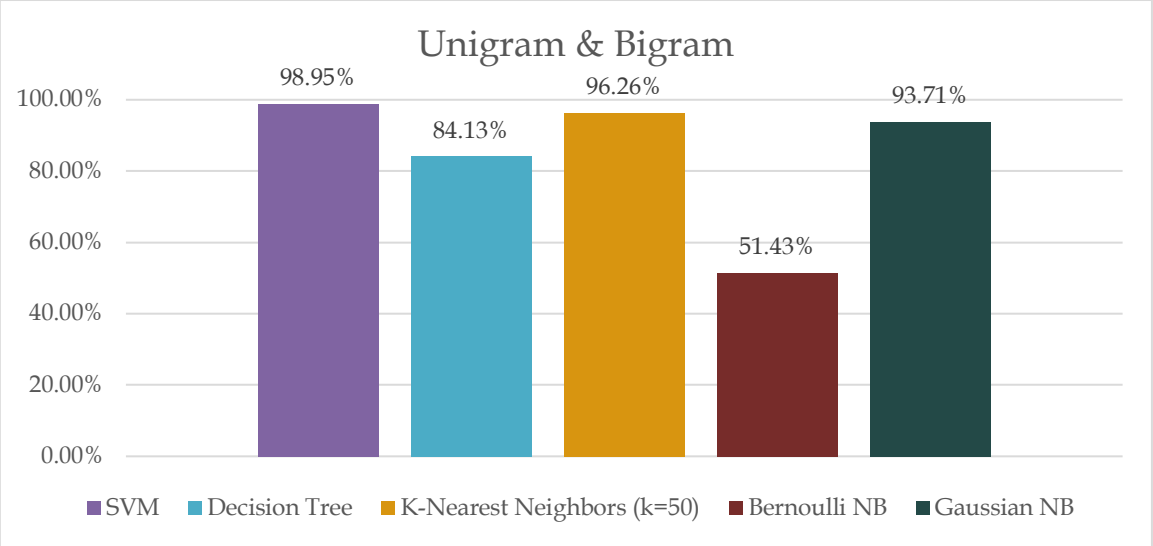

Figure 7: "Unigram & Bigram" model performances

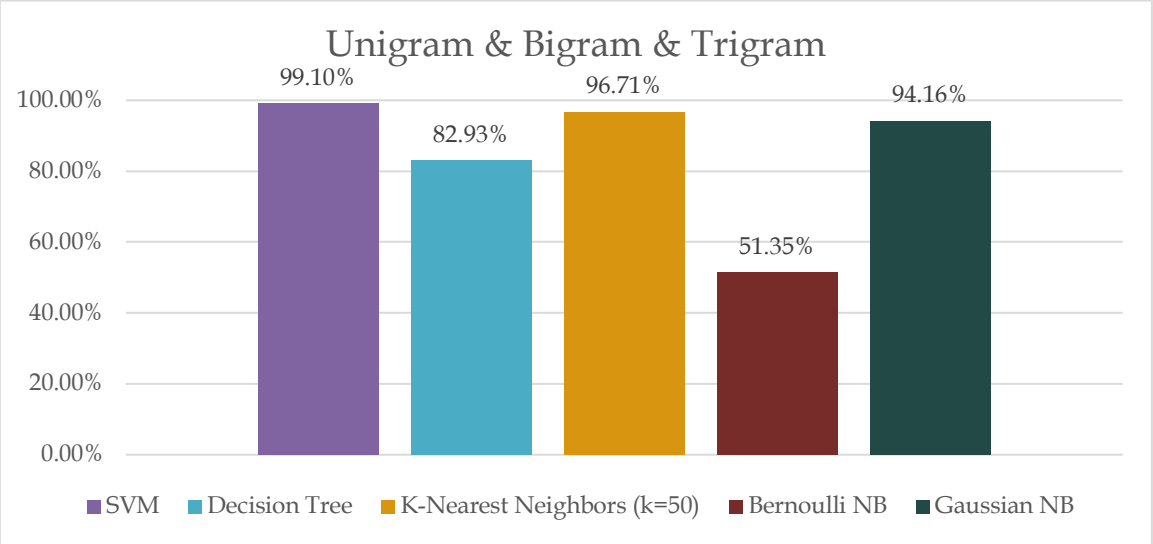

Figure 8: "Unigram & Bigram & Trigram" model performances

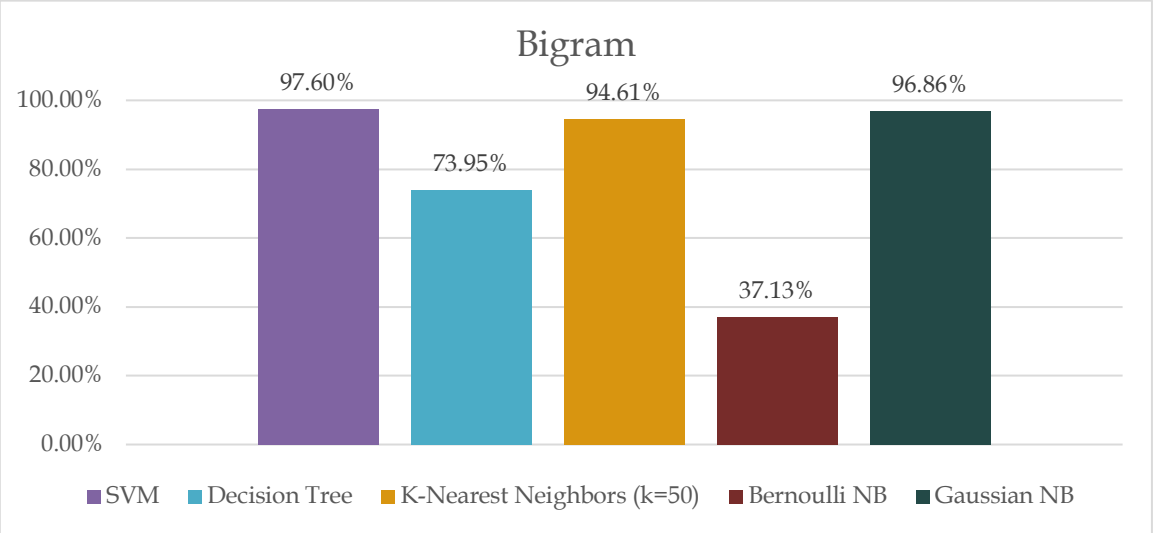

Figure 9: "Bigram" model performances

Appendix B
Postman Collection

A Postman Collection for testing the classifier's API created and exported:

```
{
   "info": {
       "_postman_id": "33d531cf-2467-4e07-8ae3-40f270292a59",
       "name": "Url Classifier",
       "schema":
"https://schema.getpostman.com/json/collection/v2.1.0/collection.json"
   },
   "item": [
       {
           "name": "Parse URL",
           "request": {
               "method": "POST",
               "header": [
                   {
                       "key": "Content-Type",
                       "value": "application/json",
                       "type": "text"
                   }
               ],
               "body": {
```

```
                    "mode": "raw",
                    "raw": "{\n    \"url\": \"https://docs.microsoft.com/en-
us/azure/cognitive-services/bing-web-
search/quickstarts/python\",\n    \"include_raw_data\":
false,\n    \"classification_model_id\": [\"general_bbc_topics\",
\"blogtext_2004_age_range\"]\n}",
                    "options": {
                        "raw": {}
                    }
                },
                "url": {
                    "raw": "{{base_url}}/parse-url/",
                    "host": [
                        "{{base_url}}"
                    ],
                    "path": [
                        "parse-url",
                        ""
                    ]
                }
            },
            "response": []
        },
        {
            "name": "Web Search",
            "request": {
                "method": "POST",
                "header": [
                    {
                        "key": "Content-Type",
                        "value": "application/json",
                        "type": "text"
                    }
                ],
                "body": {
                    "mode": "raw",
                    "raw": "{\n    \"search_query\":
\"Albert\",\n    \"specific_sites\": [\"theguardian.com\",
\"nytimes.com\"],\n    \"classification_model_id\": [\"b523c98b-07ca-4596-
8380-11b6f464bf3b\"],\n    \"page\": 1,\n    \"items_per_page\": 5\n}"
                },
                "url": {
                    "raw": "{{base_url}}/parse-web-results/",
                    "host": [
                        "{{base_url}}"
                    ],
                    "path": [
                        "parse-web-results",
                        ""
                    ]
                }
            },
            "response": []
        },
        {
```

```
            "name": "Parse URL List",
            "request": {
                "method": "POST",
                "header": [
                    {
                        "key": "Content-Type",
                        "type": "text",
                        "value": "application/json"
                    }
                ],
                "body": {
                    "mode": "raw",
                    "raw": "{\n    \"url_list\":
[\n        \"http://gingernatalie.com/2020/03/how-to-be-
happier/\",\n        \"http://teenlibrarian.co.uk/2020/11/27/4-reasons-
verse-novels-are-awesome-by-lucy-
cuthew/\",\n        \"http://teenlibrarian.co.uk/2020/10/28/three-comic-
books-from-street-noise-
books/\",\n        \"http://teenlibrarian.co.uk/2020/10/26/guest-post-a-
monument-to-cognitive-dissonance-by-lindsay-k-
bandy/\",\n        \"http://teenlibrarian.co.uk/2020/07/15/when-stars-are-
scattered/\",\n        \"http://teenlibrarian.co.uk/2008/11/04/november-is-
nanowrimo/\"\n    ],\n    \"classification_model_id\":
[\"general_bbc_topics\", \"blogtext_2004_age_range\"]\n}"
                },
                "url": {
                    "raw": "{{base_url}}/parse-urls/",
                    "host": [
                        "{{base_url}}"
                    ],
                    "path": [
                        "parse-urls",
                        ""
                    ]
                }
            },
            "response": []
        },
        {
            "name": "Upload Training Dataset",
            "request": {
                "method": "POST",
                "header": [
                    {
                        "key": "Content-Type",
                        "value": "application/json",
                        "type": "text",
                        "disabled": true
                    }
                ],
                "body": {
                    "mode": "formdata",
                    "formdata": [
                        {
                            "key": "dataset_file",
```

```
                        "type": "file",
                        "src": []
                    },
                    {
                        "key": "content_column",
                        "value": "text",
                        "type": "text"
                    },
                    {
                        "key": "label_column",
                        "value": "age_range",
                        "type": "text"
                    },
                    {
                        "key": "ngram_range",
                        "value": "1-4",
                        "type": "text"
                    },
                    {
                        "key": "algorithm",
                        "value": "SVM",
                        "type": "text"
                    },
                    {
                        "key": "algorithm_parameters",
                        "value": "",
                        "type": "text",
                        "disabled": true
                    },
                    {
                        "key": "eliminate_stopwords",
                        "value": "true",
                        "type": "text"
                    },
                    {
                        "key": "stem_words",
                        "value": "false",
                        "type": "text"
                    }
                ]
            },
            "url": {
                "raw": "{{base_url}}/upload-training-dataset/",
                "host": [
                    "{{base_url}}"
                ],
                "path": [
                    "upload-training-dataset",
                    ""
                ]
            }
        },
        "response": []
    },
    {
```

```
            "name": "Check Classification Model Status",
            "request": {
                "method": "POST",
                "header": [],
                "body": {
                    "mode": "raw",
                    "raw": "{\n    \"classification_model_id\":
\"blogtext_2004_age_range\"\n}",
                    "options": {
                        "raw": {
                            "language": "json"
                        }
                    }
                },
                "url": {
                    "raw": "{{base_url}}/get-model-status/",
                    "host": [
                        "{{base_url}}"
                    ],
                    "path": [
                        "get-model-status",
                        ""
                    ]
                }
            },
            "response": []
        }
    ],
    "event": [
        {
            "listen": "prerequest",
            "script": {
                "id": "d70bed9e-e8b3-4c76-a43f-0f886aee16ad",
                "type": "text/javascript",
                "exec": [
                    ""
                ]
            }
        },
        {
            "listen": "test",
            "script": {
                "id": "9167fd83-7658-4c20-8283-f8ea511ec082",
                "type": "text/javascript",
                "exec": [
                    ""
                ]
            }
        }
    ],
    "variable": [
        {
            "id": "657f1d32-6b89-43d0-9769-a967f516e1c7",
            "key": "base_url",
            "value": "http://127.0.0.1:8099"
```

```
        }
    ],
    "protocolProfileBehavior": {}
}
```

Appendix C
MetadataWrapper Class

JsonLdParser:

```
import json
from scrapy.selector import Selector


class JsonLdParser:

   def __init__(self, html_document=None, json_ld={}):
       self.json_ld = {}

       if type(json_ld) is dict:
           self.json_ld = json_ld

       # Extract JSON+LD metadata from HTML:
       if html_document:
           try:
               self.json_ld =
json.loads(Selector(text=html_document).xpath('//script[@type="application/
ld+json"]//text()').extract_first())
           except Exception:
               pass

       # Merge meta nodes if the root node is a list:
```

```
        self.__merge_list_nodes()

    def to_dict(self):
        return self.json_ld

    def get(self, prop):
        return self.json_ld.get(prop)

    def get_node(self, prop):
        node_obj = self.json_ld.get(prop) or {}
        return JsonLdParser(json_ld=node_obj)

    def get_first_node(self, prop):
        node_obj = self.json_ld.get(prop) or {}
        if not node_obj:
            return JsonLdParser()

        if type(node_obj) is list:
            node_obj = node_obj[0]

        return JsonLdParser(json_ld=node_obj)

    def __merge_list_nodes(self):
        if type(self.json_ld) is not list:
            return
        json_ld_list = self.json_ld
        merged = {}
        for json_ld in json_ld_list:
            if type(json_ld) is dict:
                merged = {**merged, **json_ld}
        self.json_ld = merged
```

MetadataWrapper gives a universal output of the HTML metadata combined with the JSON-LD information. Several fallbacks and keywords merging offering a simple and universal output:

```
import metadata_parser
import re
from .json_ld_parser import JsonLdParser


class MetadataWrapper:

    def __init__(self, html_document):
        self.html_document = html_document

        # Extract meta tags:
        self.meta = metadata_parser.MetadataParser(html=html_document,
search_head_only=False)

        # Extract JSON+LD metadata:
        self.json_ld = JsonLdParser(html_document=html_document)

    @property
```

```
    def raw(self):
        return {
            'json_ld': self.json_ld.to_dict(),
            'meta_tags': self.meta.metadata
        }

    @property
    def title(self):
        return (self.meta.get_metadatas('title') or [None]).pop()

    @property
    def description(self):
        return (self.meta.get_metadatas('description') or [None]).pop()

    @property
    def multimedia(self):
        return {
            'image': self.image,
            'video': self.video,
            'tags': self.__merge_keywords(self.video_tags, self.image_tags)
        }

    @property
    def image(self):
        return (self.meta.get_metadatas('image') or self.meta.get_metadatas('thumbnail') or
[None]).pop()

    @property
    def video_tags(self):
        return self.meta.get_metadatas('video:tag') or []

    @property
    def video(self):
        return (self.meta.get_metadatas('player') or self.meta.get_metadatas('video:url') or
[None]).pop()

    @property
    def image_tags(self):
        return self.meta.get_metadatas('image:tag') or []

    @property
    def content_type(self):
        return (self.meta.get_metadatas('type') or [None]).pop()

    @property
    def locale(self):
        return self.meta.get_metadatas('locale')

    @property
    def site_name(self):
        return (self.meta.get_metadatas('site_name') or
self.meta.get_metadatas('al:android:app_name') or
self.meta.get_metadatas('app:name:googleplay') or
self.meta.get_metadatas('al:iphone:app_name') or [None]).pop()

    @property
    def keywords(self):
        return (','.join(self.meta.get_metadatas('keywords') or []).replace(' ',
'')).split(',')

    @property
    def news_keywords(self):
        return (','.join(self.meta.get_metadatas('news_keywords') or []).replace(' ',
'')).split(',')

    @property
    def all_keywords(self):
        return self.__merge_keywords(self.keywords, self.news_keywords)
```

```
    @property
    def modified_time(self):
        return self.json_ld.get('dateModified') or (self.meta.get_metadatas('updated_time')
or self.meta.get_metadatas('article:modified_time') or
self.meta.get_metadatas('article:modified') or [None]).pop()

    @property
    def published_time(self):
        return self.json_ld.get('datePublished') or self.json_ld.get('uploadDate') or
(self.meta.get_metadatas('article:published_time') or
self.meta.get_metadatas('article:published') or
self.meta.get_metadatas('video:release_date') or [self.__scrap_meta_publishdate_in_js()] or
[None]).pop()

    @property
    def author(self):
        return self.json_ld.get_first_node('author').get('name') or
(self.meta.get_metadatas('article:author') or self.meta.get_metadatas('byl') or
[None]).pop()

    def __scrap_meta_publishdate_in_js(self):
        """Attention! This is fallback for the very specific case of YouTube!"""
        try:
            # Example from YouTube.com: ,\\\"publishDate\\\":\\\"2017-10-10\\\",
            publish_date_meta =
re.findall(f'",\\\\\\"publishDate\\\\\\":\\\\\\"(.+?)\\\\\\",', str(self.html_document))[0]
            if publish_date_meta:
                # TODO: Ensure that publish_date_meta is a date
                publish_date_meta.split('T')[0].split(' ')[0].strip()
                publish_date_meta += 'T00:00:00.000000'  # Add timestamp info
            return publish_date_meta or None
        except Exception:
            return None

    def __merge_keywords(self, *args, to_lower_case=False):
        merged_kw = set()
        for kw_list in args:
            for kw in kw_list:
                if kw:
                    if to_lower_case:
                        kw = kw.lower()
                    merged_kw.add(kw)
        return list(merged_kw)
```

Appendix D
Use Examples of Katigoriopoiitis

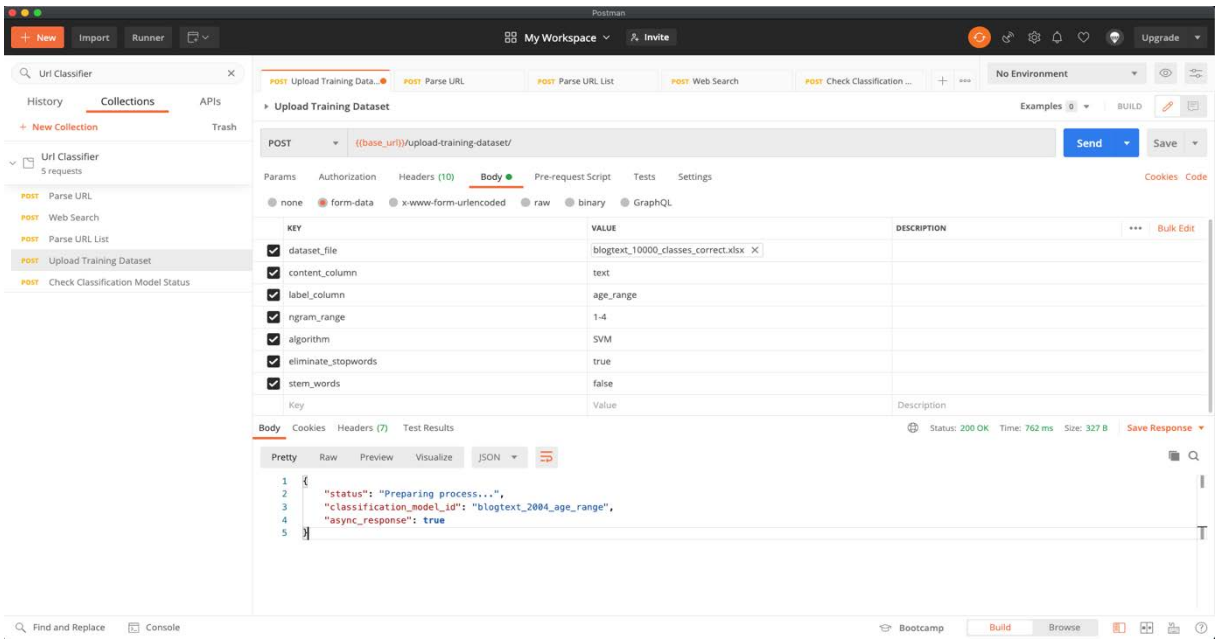

Figure 10Training model can run in the background. Classifier returns a classification model id.

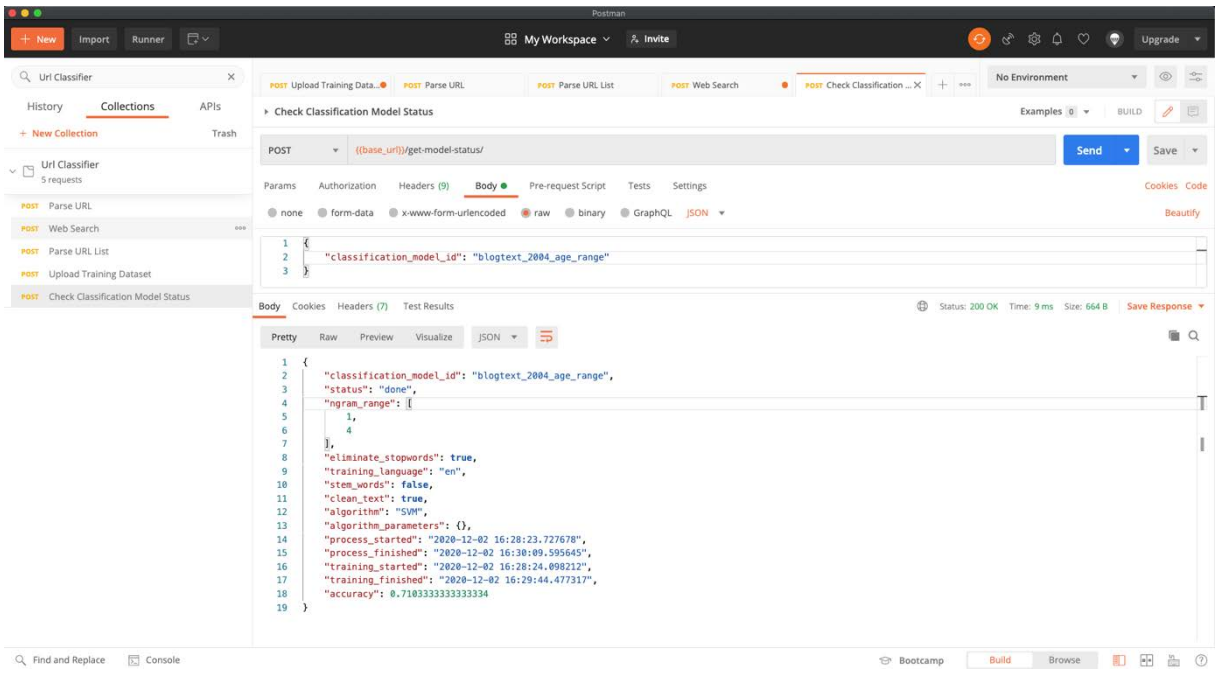

Figure 11We can check model status anytime during or after training. When the process is finished, status value turns to done. The model accuracy is calculated and included in get-model-status response.

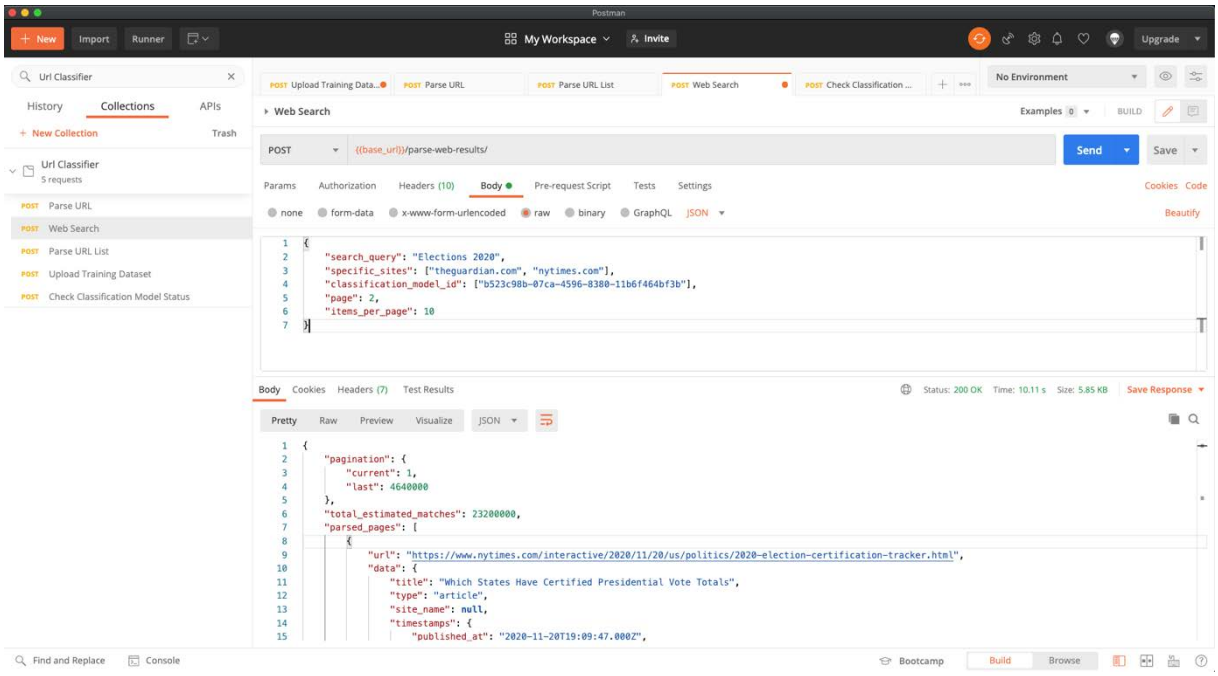

Figure 12Web search can be limited in specific websites. Pagination information included in the response.

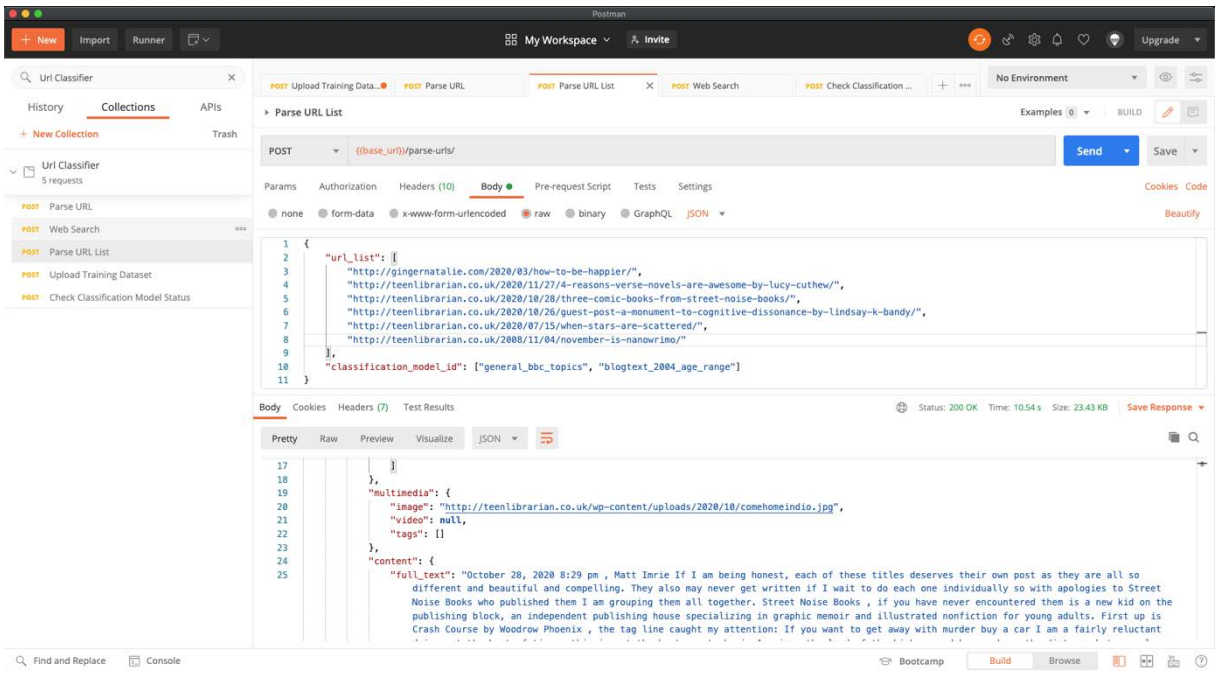

Figure 13Multiple classification models can be included in the request. The predictes_classes will be presented as an array of predictions, ordered as the model ids respectively

Example Request:

```
{
   "url": "http://teenlibrarian.co.uk/2020/07/15/when-stars-are-scattered/",
   "include_raw_data": false,
   "classification_model_id": ["general_bbc_topics", "blogtext_2004_age_range"]
}
```

Example Response:

```
{
   "url": "http://teenlibrarian.co.uk/2020/07/15/when-stars-are-
scattered/",
   "data": {
       "title": "When Stars are Scattered",
       "type": "article",
       "site_name": "Teen Librarian",
       "timestamps": {
           "published_at": "2020-07-15T08:30:00+00:00",
           "updated_at": "2020-07-12T09:43:41+00:00",
           "parsed_at": "2020-12-02T21:27:28.276194"
       },
```

```
        "locale": {
            "lang": "en",
            "tags": [
                "en_US"
            ]
        },
        "multimedia": {
            "image": "http://teenlibrarian.co.uk/wp-
content/uploads/2020/07/BlogTourLowRes.jpg",
            "video": null,
            "tags": []
        },
        "content": {
            "full_text": "July 15, 2020 9:30 am , Caroline Fielding Omar and
his brother Hassan, two Somali boys, have spent most of their lives in
Dadaab, a refugee camp in Kenya. Separated from their mother, they are
looked after by a friendly stranger. Life in the camp isn't always easy and
the hunger is constant . . . but Omar devotes everything to taking care of
his young brother and pursuing his education. Faber This is set to be one
of my favourite graphic novels of all time. You will laugh, cry, rage, and
cheer many times over the course of the book, a study in empathy, as Omar
and Hassan experience the ups and downs of life in a refugee camp with the
dream of resettling in America hanging over their heads. It is based on
Omar Mohamed's account of real experiences of growing up, so obviously the
relationships are real, but they are brought off the page so beautifully
and in so few words, through the skillful work of Victoria Jamieson
(brilliantly coloured by Iman Geddy). Narrated by Omar, we see his
perspective of the environment and people, and how it changes when he was
feeling hopeful or down. Bad things do happen to them, as well as good
things, and Omar talks them through and shares his feelings with the
reader. One panel that really struck me was after Omar had been talking to
a friend who's family had been chosen to be resettled, he tries so hard to
be positive all the time but can't help but think "It's not fair". He tells
us: ...Of course, thinking like this doesn't do you any good. Somalis even
have a word for it. BUFIS. It means the intense longing to be resettled.
It's almost like your mind is already living somewhere else, while your
body is stuck in a refugee camp... We first meet Omar and his brother
Hassan once they have already been living in the camp for a long time (have
a read of the first chapter in the extract) and the way their journey to
the camp is told to us, as it recounted in Omar's UN interview for
potential resettlement, is really powerful. We follow them for years, until
Omar is 18, and I was particularly moved by the relationship with Fatuma,
how they came to be together, and how Omar realised more and more with age
how lucky they all were to have one another. Enjoy this exclusive extract
of WHEN THE STARS ARE SCATTERED It does have a happy and hopeful ending for
Omar and Hassan, but doesn't let you forget the thousands more people still
stuck in the limbo of refugee camps. I think this is essential reading for,
well, everyone aged 8+ frankly. Huge thanks to Faber for sending me a copy
for review and inviting me to join the blog tour. WHEN STARS ARE SCATTERD
is out in the UK now! Share this:",
            "author": null,
            "tags": [],
            "keywords": [
                ""
            ],
```

```
            "news_keywords": [
                ""
            ],
            "extract_code": "1_1"
        },
        "errors": [],
        "predicted_classes": [
            "entertainment",
            "23-27"
        ]
    }
}
```