ART TICKER - DISCOVERING EMERGING ARTISTS ON THE WEB

BY

SAAD PATEL

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Computer Science

Written under the direction of

Tomasz Imielinski

and approved by

_____

_____

_____

New Brunswick, New Jersey

January, 2016

ABSTRACT OF THE THESIS

ART TICKER - DISCOVERING EMERGING ARTISTS ON THE WEB

by Saad Patel

Thesis Director: Tomasz Imielinski

Considering the large number of artists that exist, there is valuable talent to be discovered. But the question arises, how to find promising and emerging artists from hundreds of thousands of names listed among many different aggregations websites such as artfacts.org, and thousands of art galleries? We introduce an application named ArtTicker which uses many features of Machine Learning, Information Retrieval, Data Mining and Text Mining to crawl, rank, and analyze artists and their popularity on the web.

We start by identifying names of artists who are not yet listed in large aggregate directories (such as artfacts) but are already represented by some galleries. This task requires crawling and extraction of artist names from thousands of art galleries. These web sites share a lot of common structures, however there is also significant variety among them and artist name extraction requires complex heuristics. We harvest thousands of artist names this way. Then we enter the second phase of the project ranking this artists by their web presence.

Since the wealth of any data mining model is the actual data, the data collection period consisted of extensive crawling from a vast number of news publication websites. To this end we gather and cluster news from several leading art related news websites and also use many signals to rank and classify these art news sources. The artistss score is based on how significantly an individual artist was featured in the art news stream of articles. The final objective of finding the emerging artists is met by identifying the names which are present on gallery web sites, have high media presence (high score) and are not listed yet on the artist aggregate sites. The working prototype analyzes over 150 sources in English language but can be easily extended based on automatically crawling and analyzing related sources. It currently holds over 250,000 artists and over 70,000 articles from all these news sources. In essence, this is a streaming application for which given any geographic area (say Lower Manhattan) identifies the hottest artists who are not yet known.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Prof. Tomasz Imielinski for the continuous support of my Masters study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Masters study.

I would like to thank my family: my parents and to my brothers and sister for supporting me spiritually throughout writing this thesis and my my life in general.

Dedication

This thesis is dedicated to my parents for their constant support and invaluable encouragement.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Art market is like stock market only less predictable. Who can be the next star commanding high prices for their work and who will disappear into oblivion? This is far more difficult to predict than raises and falls of stocks. But rewards may be much larger. Damien Hirst sells for tens of millions of dollars and he is in his late forties, Jeff Koons is even more expensive - and they raised to prominence within a decade or two

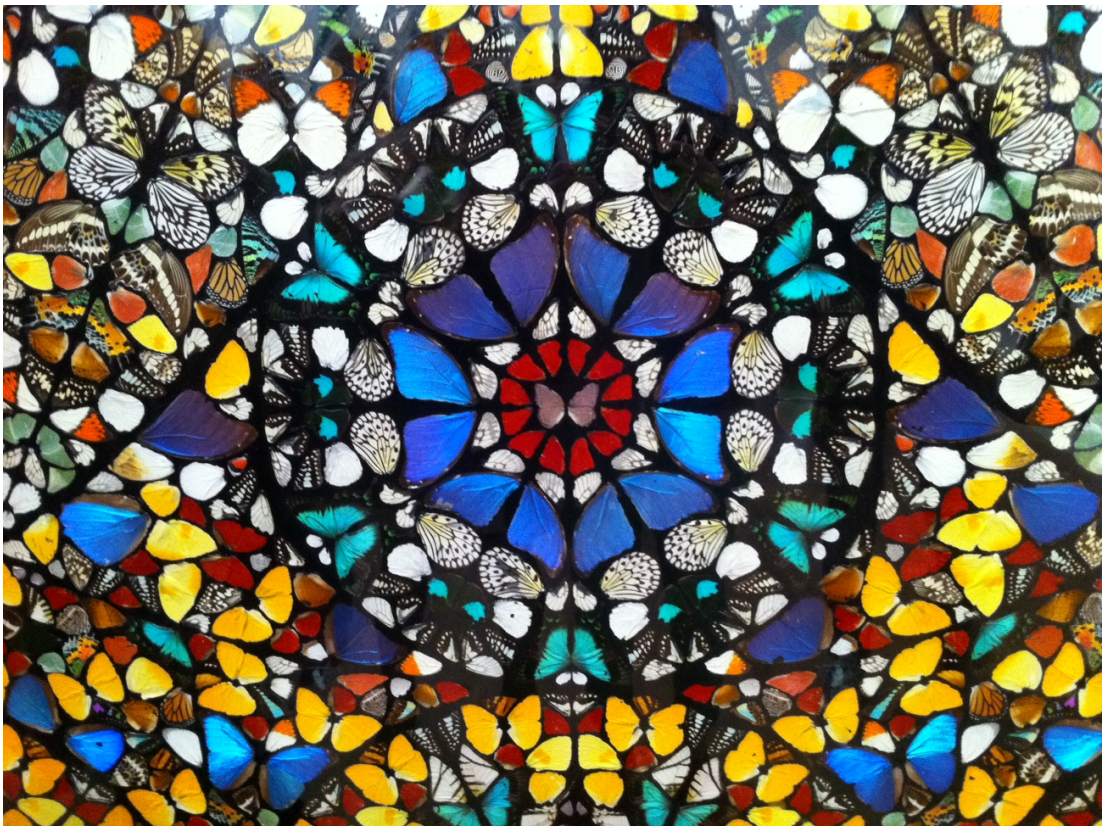


Figure 1.1: Artwork of Damien Hirst

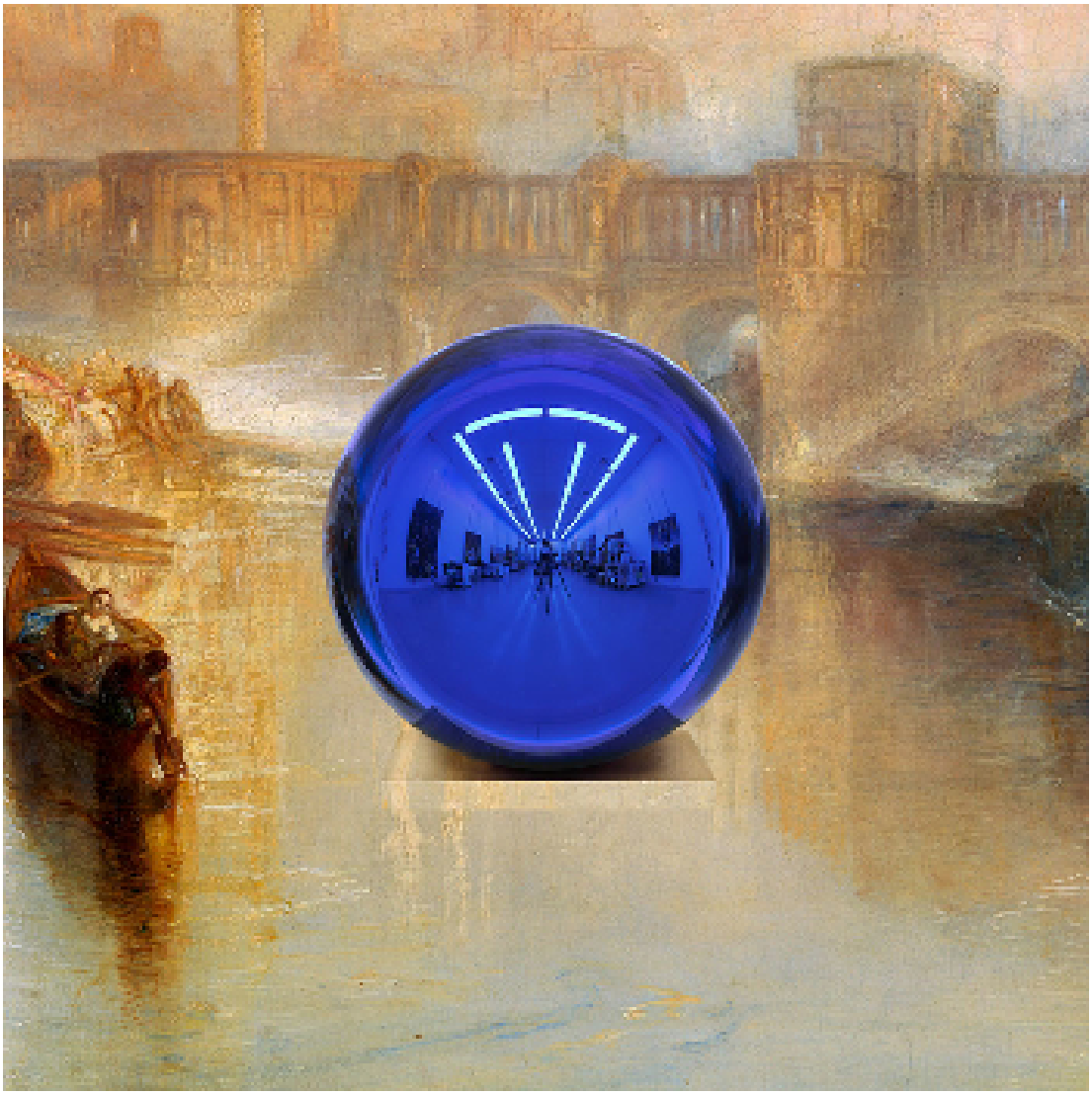


Figure 1.2: Artwork of Jeff Koons

Five years ago Oscar Murillo and Parker Ito were struggling and unknown painters. Today they command six figure prices for their paintings and are represented by top galleries. They have already emerged as young stars of their generation. Is it possible to capture such meteoric rise early, before it happens?

Figure 1.3: Artwork of Oscar Murillo

Figure 1.4: Artwork of Parker Ito

It seems next to impossible, but with the web and social media it is probably easier than decades ago.

The objective of this thesis is to use web signals and online media presence to provide art collectors with such early signals. The lead intuition is that all emerging artists are first "talked about" at various artistic blogs and small art magazines before they reach major media (when they become famous). For example an article in small artsy publications such as Hyperallergic or Brooklynrail about an artist could be an indication to put someone on the radar screen of a collector.

But monitoring hundreds or even thousands of such art publications would require a lot of time. One alternative would be to consult aggregators such as artfacts.net which provide ranking of 150,000 plus artists based on rich information about exibitions they participate in and whom do they exibit with in group shows. Unfortunately sources like this, although useful in longer term, are stale, and fall behind due to being updated manually. It takes a long time to climb the rankings of artfacts so even young stars such as Parker Ito and Oscar Murillo still have not cracked top 1000 painters (occupied by old masters and Picasso, Jackson Pollock, Rothko alikes).

We propose a different approach which has two stages. In the first one we crawl and extract names of artists from thousands of galleries (Thus, for us to consider an artist, s/he has to be represented by a gallery). Then we check the ArtTicker news feed which have built from hundreds of RSS of news content sites and built a Media Score of an artist. We are very interested in artists who are not yet represented in artfacts rankings and are already generating media buzz. Using our Media Score (which takes under consideration the prestige of the source and the degree of presence of artist in the article), we rank new artists from most media present to the least. In ArtTicker

system, it is performed geographically, a user virtually visits a city, or a location (say Chelsea in Manhattan) and then we scrape names of artists from nearby galleries and rank them by media score.

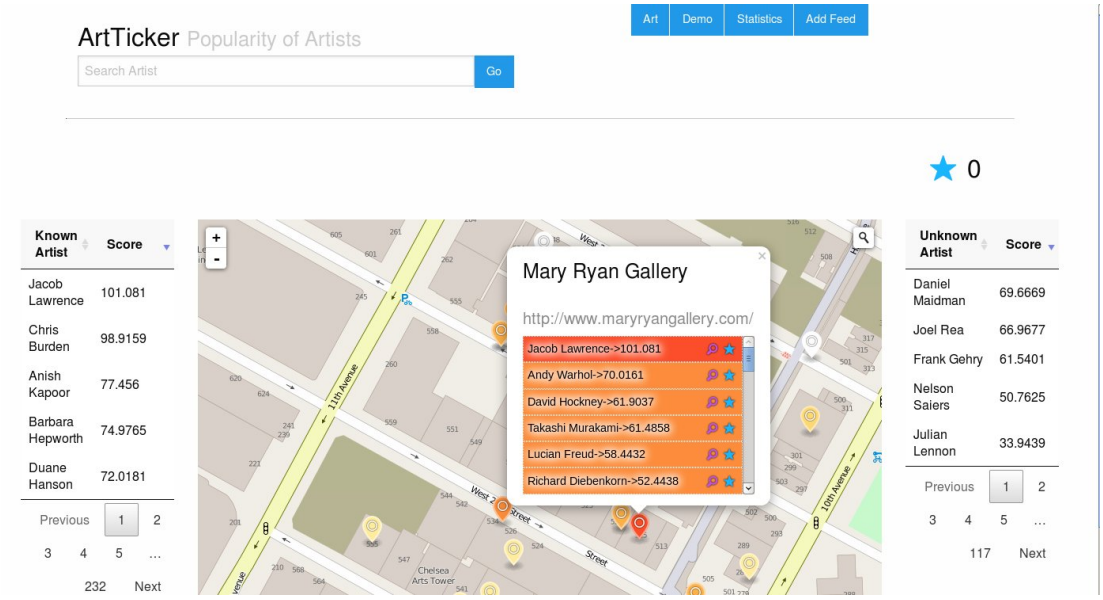


Figure 1.5: Demo of ArtTicker

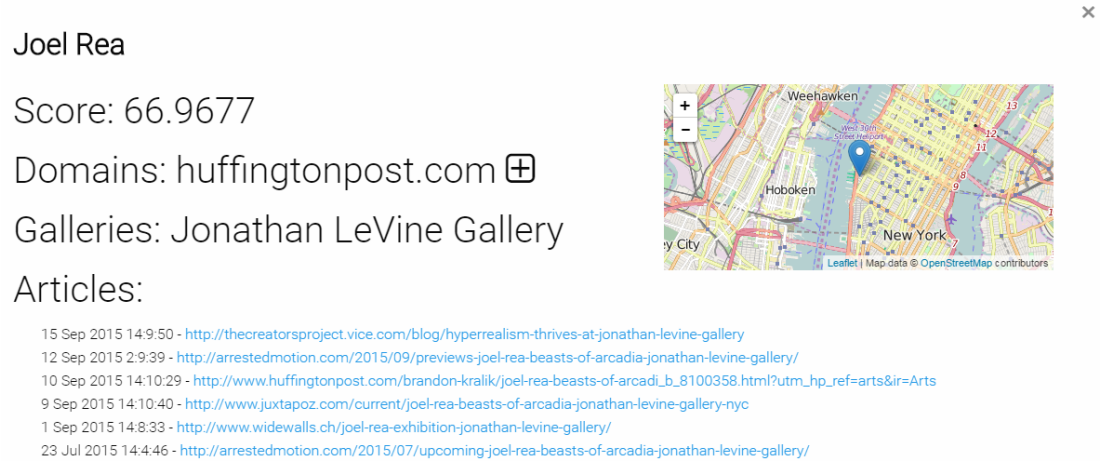


Figure 1.6: Demo of ArtTicker Artist Ranking

1.2 Objective

As many artists are increasingly getting featured on the Internet, the number of articles and quality of articles on them are also increasing. Many times this information is too much average user. This dissertation aims at accumulating and clustering the massive amount of information present on the web on artists and present useful insights namely, finding "Hottest" Artists and their representations in local galleries. The major emphasis is to aggregate enough data on art and artists and implement an algorithm which will automatically rank the artists based on the extracted information from each article and extract new artists from local galleries. The objective is to present an impartial view of the collection of information. The following include some of the important contributions of this dissertation:

- A name extraction algorithm based on the current state of web pages and primarily focusing on artists represented in local galleries.
- Creating relationships between known and unknown artists and relevancy of the articles in which they are mentioned along with the sources of those articles.
- Building models based on the relationships which show top artists for each gallery and the inverse mapping of top artists which are mentioned in a particular gallery.
- Defining a ranking model for artists, which is based on the amount of attention the sources receives

We represent the procedure for the above in Figure 1.

1.3 Structure

In Chapter 2 of this thesis, we first present background of all the research areas related to the application such as: Information Retrieval, Text Mining, Data Mining, Machine

Learning, and the algorithms used in the application. We describe the TF-IDF, SVM, Classification and how they really work. In Chapter 3, we discuss how the data was gathered for this research, and the data transformations that were done on the data-set. In Chapter 4, we discuss the database model and how the data is continually loaded to the database. In Chapter 5, we show the different techniques for Article Classification and analyze each technique and its accuracy. In Chapter 6, we describe an algorithm for Artist Name Extraction and analyze the accuracy of it. In Chapter 7, we describe how we assign ranking to the sources and articles use that to determine the ranking of the individual artists. Finally, we discuss about future work and conclusion in Chapter 8.

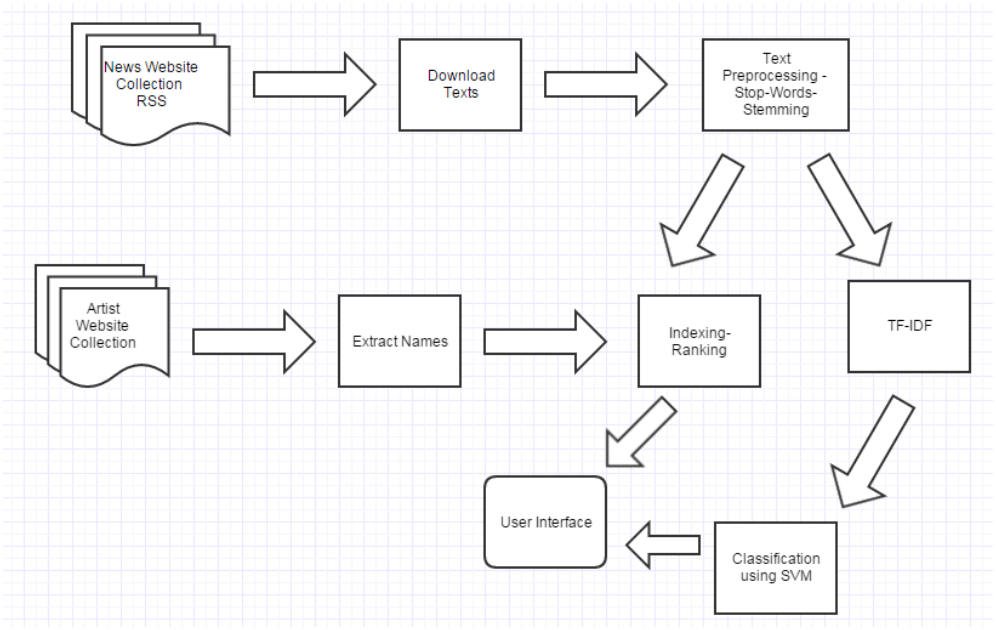


Figure 1.7: How Art Ticker algorithm works

Chapter 2

BACKGROUND

In this chapter we describe all the main concepts of the research areas that were used in this paper. The main research areas used include: Information Retrieval, Text Mining, Data Mining, Machine Learning.

2.1 Information Retrieval

2.1.1 Text Mining Algorithm

One of the main tasks is analysis and classification of text. To do this, one has to find all the words and their rankings in a document and further use that with a classifier. For the first task of ranking words in a corpus of documents, one of the most trusted and widely used representation for this task is TF-IDF which we will introduce below.

Introduction to TF-IDF

TF-IDF is defined as term frequency inverse document frequency. It used widely in information retrieval and text mining. In simple terms it is a numerical statistical measure that is used to determine how important a word-term is to a document in a collection of documents or a corpus. The importance of a word increases proportionally to the number of times a word appears in a single document vs multiple documents. TF-IDF is used in many fields including search engines to rank a documents relevance to a query given by the user. Furthermore it is also used in stop-words filtering, in text

summarizing and classification. In short, TF-IDF works by calculating the frequency of words in one document and compares it to the inverse ratio of the word over the entire list of documents. The TF-IDF determines how relevant a given word is in a certain document. The benefit is that common words in English such as such as articles and prepositions also known as stop words get a lower score than words which are not very common and don't occur in many documents. This is primarily the main reason that it is used in matching the relevancy of a user's query to a group of documents.

How TF-IDF Works

To start, we suppose we have a group of documents and we wish to determine which document is the most relevant to the query x. The first step is to remove all the documents that do not contain any of the words in query x. This, however still leaves many documents from the corpus. The next step is to count the term frequency which is defined as the number of times each term occurs in each document.Wen Zhang [2008] Thus we get the following formula which depicts the term frequency:

$$tf(t,d) = f_{t,d}$$

Where $tf(t,d)$ is the term frequency given a document and $f_{t,d}$ is the raw frequency of the term t in document d.

We can see that many stop words such as "the" and "is" are common and occur in many documents given a list of documents with high frequency because of their commonality. This will increase the importance of the documents which use this word frequently and will decrease the weight of importance of the other words in the query. Therefore, an inverse document frequency factor is used which is designed to lessen the

weight of terms that occur very frequently in many documents and instead increases the weight of terms that occur rarely. This is referred to as inverse document frequency.Ventouris [2014]

Another issue that still exists, is that each document has a different length, therefore larger documents will get higher preference than a smaller document. To fix this the document needs to be normalized and this is done by dividing the term frequency by the total number of terms in the document. Furthermore, the TF-IDF score also takes into account the heuristic that words that appear many times in a documents are more likely to be important than words that appear once. To do this we apply a logarithmic weighting. Thus for the inverse document frequency we get the follow formula:

$$idf(t, D) = log(\frac{N}{f_{t,D}})$$

Thus we have the following equation:

$$TF * IDF(t, d, D) = f_{t,d} * log(\frac{N}{f_{t,D}})$$

Where $f_{t,d}$ is the number of times term t appears in document d, divided by the total count of terms in document d, N is the size of all the documents, and $f_{t,D}$ represents the number of documents in which w appears in D. Ramos [2003]

2.1.2 Extraction Algorithm

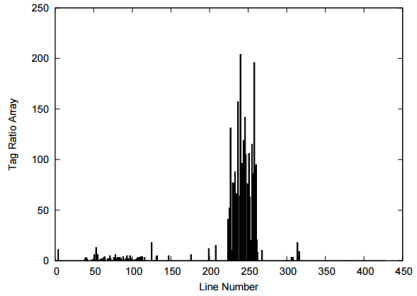
One very important algorithm that we rely on in this thesis is given the raw HTML of a website, we would like to analyze it and extract the main content. This is also referred to as boilerplate detection. To do this we use a variety of different algorithms with different features for each algorithm. In the end we use all these features and use machine learning on it for maximum usefulness. Peters and Lecocq [2013]

Feature 1 - Shallow Features(ST)

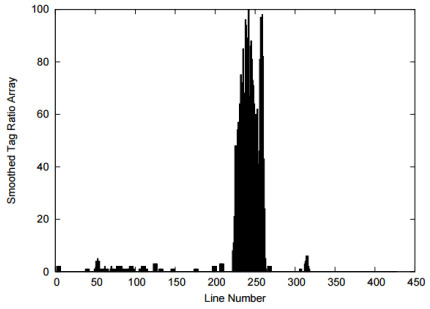
The first set of features which we would like to include are shallow text features. The content of a particular web page can be divided into two groups, one with tags which hold long strings of text such as paragraphs represented by $< p >$ tag. And the second group can be tags with short text such as advertisements and navigational elements on a web page. We can therefore use text density which encompasses average word length, average sentence length, and link density of tags as well as the textual density of the tags surrounding it as features. Christian Kohlschtter [2010]

Feature 2 - Tag Ratio(CETR)

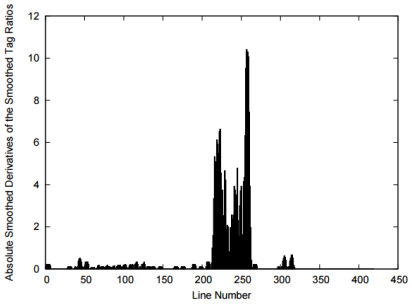
Another set of features we can use is the tag-ratio. In other words the ratio of number of tags in a specific block to the text length. These set of features have been known to be very effective in extracting content of the web page. The detection works by the obvious assumption that the main content of a given web page is clustered together while other elements such as navigational elements and advertisements are present throughout web page without much clustering. Then for each line in the HTML document the algorithm computes the text to tag ratio and represents it in a graph. The next step is to smooth the graph using Gaussian smoothing or any other smoothing algorithm so that the textual content can be more easily discerned. We can see that the beginning and end of the content are represented by sharp rises in the graph. Therefore, the last step is to determine the start and end of the content, which can be done by taking the absolute derivative of the smoothed graph and plot it against the smoothed tag ratio. The first image on the left below represents the text-to-tag ratios of the web page line by line. The second shows the smoothed graph of the text to tag rations. The third image shows the absolute derivative of the smoothed graph. The last images shows the absolute smoothed derivatives of the smoothed tag ratios. Tim Weninger [2010]



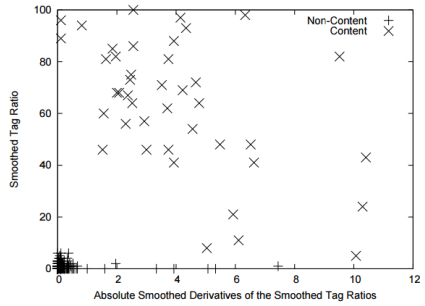
(a) Tag Ratios of a Sample Web Page



(b) Smoothed Tag Ratios of a Sample Web Page



(a) Absolute Derivative of Smoothed Tag Ratios of a Sample Web Page



(b) Absolute Smoothed Derivatives of the Smoothed Tag Ratios of a Sample Web Page

As we can see in the last image, the non-content are clustered around the origin of the graph while the content-areas are scattered around the graph. Using this technique, the accuracy of determining the content of a web page using the F1 score metric is around 80%.

Feature 3 - Tag Classes(IC)

The third set features which we can include to detect textual content of a given web page is to use the *id* and *class* attributes in HTML. The *id* and *class* attributes are used by developers to identify tags in the HTML of a website. Many times it can include words such as ”content”, ”navigation”, ”menu” to indicate different places in the

HTML document. We can use these attributes from training data to see which which ones occur the most. We can then use a cut-off minimum value for attributes in tags of actual textual content vs non-textual content and we can use these as features for the tags. Peters and Lecocq [2013]

Based on all these approaches we use a metric called the F1 score which is a widely used measure of a test's accuracy. It is defined as:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Based on this the result of the combination of the features is as follows:

Table 2.1: Content Extraction Results

Method	F1 Score
Baseline	62.5%
CETR	79.4%
IC	64.1%
ST	85.4%
ST+IC	85.8%
ST+IC+CETR	87.7%

Peters and Lecocq [2013]

2.2 Machine Learning

2.2.1 Theory of Support Vector Machines

Introduction to SVM

In the following chapters we would like to classify texts given a large corpus of text and we would like to classify it into defined categories based on the content of the next. This task can be accomplished with SVM(Support Vector Machine Algorithm). Text mining algorithms are generally divided into two types. One employs supervised learning while the other uses the unsupervised learning model. Support vector machines (SVMs) are based on a set of supervised learning methods used for classification which we we delve into. In short, with SVM, text documents are classified based on a linear or nonlinear maximal separating hyperplane derived from a given input training data set.

How SVM Works

We will explain the SVM algorithm graphically as to make it easier to understand. To start, we are given as training data a list of features and labels for the features and we would like to classify unseen features based on the training data. This is known as the test data. To start we will first introduce a simplification to the SVM which is named Perceptron.

Perceptron

We start with the training data and we would like to classify them and use it to predict classifications for future test data. For the illustration we assume that the data is in 2 dimensions and we use a binary classifier. We start by first plotting the features on a grid and we try to separate them using the perceptron algorithm. We have the following representations:

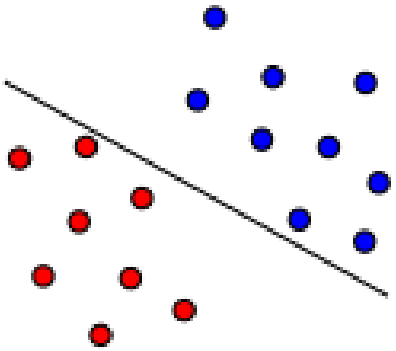


Figure 2.3: Possible Perceptron Solution 1

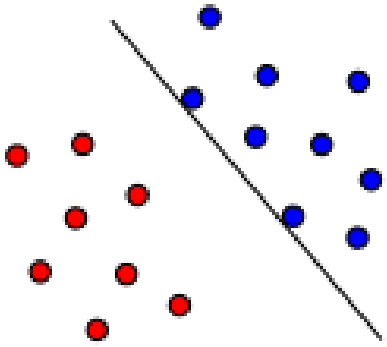


Figure 2.4: Possible Perceptron Solution 2

As we can see in Figure 2.1 and Figure 2.2 above that given features plotted in a graph with labels, the perception classifier finds any line or in higher dimensions any hyperplane which linearly separates the data. The downside of this algorithm is that it doesn't necessarily find the best line which divides the classes but rather it finds a possible line. It's benefit is that it allows online learning because of its approach in processing one element at a time from the training set. Formally we define perceptron algorithm as follows:

We are given a training set which is defined as follows: $T = (x_1, l_1), (x_2, l_2), ..., (x_k, l_k)$ with size of k. And we have each x_i is a vector of size n and each l_i to be the label for the features x_i. We initialize the weight vector which is the values for the separating hyperplane to zero. We then run the following algorithm for all the data in the training

set:

Algorithm 1: Perceptron Algorithm

Input: Training Data: $T = (x_1, l_1), (x_2, l_2), ..., (x_k, l_k)$

Output: A list of weights w which represent a line/hyperplane separating the classes

```
w ← [0, ...., 0]
for i ← 1 to k do
    if (class(w^T * x_i) ≠ l_i) then
        w ← w + l_i · x_i
return w
```

From the algorithm, we see that the algorithm finds any line which separates the different classes based on the training set and uses that to classify the test data. Depending on our data we can either continue this algorithm for all training samples, or if it used in online learning then it can be continually used until a specified error threshold is reached or it converges if the data is linearly separable.

SVM

Now that we have seen how the perceptron classifier works, we can look at the similarity and difference between that and the SVM classifier. We start with a similar situation to perceptron, where we are given as training data a list of features and labels for the features and we would like to classify unseen features based on the training data. To start, a perceptron would find any line which separates the classes. However with SVM the technique is to find the line with the maximum margin between the classes as to find the best classification line when the data is linearly separable. For the illustration we assume that the data is in 2 dimensions and we use a binary classifier. Srensen [2014] We start by first plotting the features on a grid and we try to separate the classes of the data and see we have the following representation:

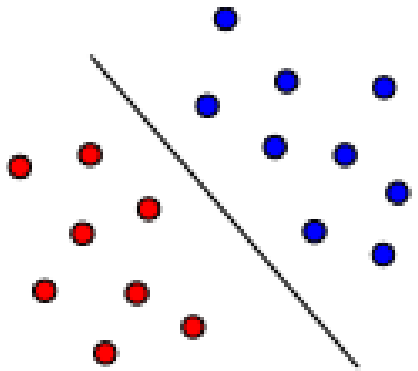


Figure 2.5: Possible SVM Solution

As we can see in the figure above, SVM finds the maximal margin line or hyperplane in higher dimensions with more features for separating the data. Specifically it finds the maximal margin line which separates the closest points from the classes. The downside of this algorithm is that it doesn't benefit from online learning as every new training data requires the whole classification algorithm to run again. This however benefits from being much more accurate than perceptron.

We can formally define it as follows: We start with the fact that there are infinite choices for hyperplane to separate two classes. To choose the best one, we need to find one which maximizes the margin between the two classes. A hyperplane is defined as:

$$\sum_i w_i x_i + b = \vec{w} \cdot \vec{x} + b = 0$$

Where $\vec{w}$ is a normal vector to the hyperplane also known as the weight vector, $\vec{x}$ is any set of points in the same dimension as the hyperplane, and b is a scalar which is referred to as the bias. This equation applies to all dimensions because as the dimension scales up the size of $\vec{x}$ and $\vec{w}$ also increase. If we suppose that bias is 0. Then we have $\vec{w} \cdot \vec{x} = 0$ which is defined as a line or hyperplane which is perpendicular to $\vec{w}$ and goes through the origin. The bias, b, determines the actual translation from the origin of the hyperplane. Furthermore if we would like to move in the direction of the vector $\vec{w}$, then

the equation: $\frac{b}{||\vec{w}||}$ represents the offset along the normalized vector $\vec{w}$. We then suppose that we have training data that is linearly separable, in other words we can plot the features of the classes and select two hyperplane between the two which separates the data without any data points between the classes. We then label the space between the hyperplanes as the margin between the two classes which we would like to maximize. To do this we find two hyperplanes which are represented by the following equations which are normalized:

$\vec{w} \cdot \vec{x} + b = 1$, $\vec{w} \cdot \vec{x} + b = -1$

Since our goal is to find the maximum margin hyperplane, we then have to include the constraint that the data points from either class cannot fall into the margin area. This is represented as follows:

$\forall i \in \{1, ..., n\} \vec{w} \cdot \vec{x_i} + b \geq 1$, and $\forall i \in \{1, ..., n\} \vec{w} \cdot \vec{x_i} + b \leq 1$ for both classes. This can be rewritten as: $\forall i \in \{1, ..., n\} class_i(\vec{w} \cdot \vec{x_i} + b \geq 1)$

Since we know that $\frac{b}{||\vec{w}||}$ represents the offset along the normalized vector $\vec{w}$, then we know that the margin between these two hyperplanes is $\frac{2}{||\vec{w}||}$. We therefore want to maximize the following term: $\frac{2}{||\vec{w}||}$, so that the distance between the two hyperplanes are maximized and we can find the best hyperplane to separate the data. We can rewrite this as minimizing $||\vec{w}||$. This however can be rewritten as: minimizing $||\vec{w}||^2$ since $||\vec{w}||$ has a square root and it is difficult to solve it. Furthermore, we add a term of $\frac{1}{2}$ for ease of calculation. We can see the representation of the margin which we want to maximize in Figure below.

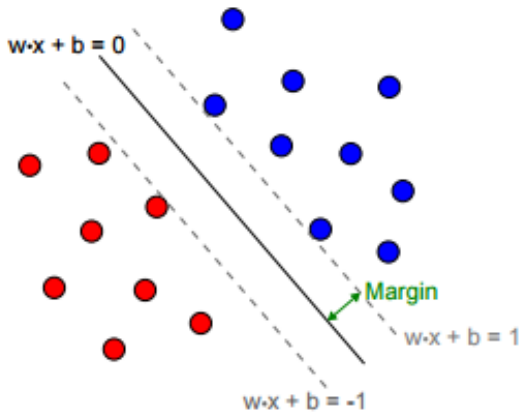


Figure 2.6: SVM Maximum Margin Classification

We then have the resulting optimization problem:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}||w||^2 \\ &\text{subject to} && \forall i \in \{1, ..., n\} class_i(\vec{w} \cdot \vec{x_i} + b) \geq 1 \end{aligned}$$

Overlapping Classes

Since the classes are not always linearly separable, and might overlap so that there can be no hyperplane which seperates them. The figure below shows an example where this can be the case.

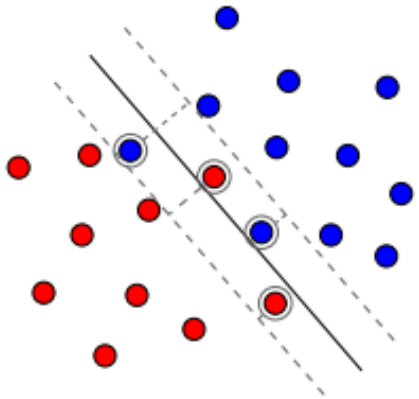


Figure 2.7: SVM Classification Overlapping Classes

We would therefore like to find a way to classify based on some slack, so that a little overlapping might not cause a problem in classification. We introduce two new variables, C which controls the trade-off between large margin and small error penalty, and ξ which is a slack variable for each point in the slack region. We have the following optimization problem after introducing these variables:

$$\text{minimize} \quad \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to} \quad \forall i \in \{1, ..., n\} class_i(\vec{w} \cdot \vec{x_i} + b) \geq 1 - \xi_i$$

In the equation, C is a scalar which controls the trade-off between classification and strictness. In the case where C is very small, the slack variables have more slack margin and this can cause misclassification. For the case where C is veyr large, this can cause the slack margin to decrease and leave very little slack.

The ξ_i in the equation represents the slack for the i-th feature vector.In the case where $\xi_i = 0$ then there is no slack and the feature vector is in the correct side of the hyper plane. In the case where $0 < \xi_i \leq 1$ then the slack of the feature vector is in between the separating hyperplane and the margin hyperplane. In the case where $\xi_i \geq 1$ then the slack of the feature vector is on the wrong side of the separating hyperplane. Another simple technique which is more widely used is to project the data into a higher dimension. This helps to make the data linearly separable when its not able to be linearly separated.

Feature Selection for Text

The features which represent a problem which we either want to predict or classify is the foundation for the accuracy of the machine learning model. For this task we are given a list of documents and we would like to transform it to a representation which is a suitable input for SVM for classification. We also use an Information Retrieval technique based on that fact that words have many different forms which prefixes and

suffixes, to stem the words and keep only the root for a better feature selection. We then have the following representation of words in terms of features: $(tf - idf(w), w)$, where the w is the numerical representation of the word and $tf - idf(w)$ is the tf-idf score for the word. This still leads to a very high number of terms and we would like to avoid a large number of features as to reduce running time of the classifier. To do this we have a cut-off α for the words which occur too little times or if the word is a stop-word. Based on this technique we classify documents using the SVM classifier based on their terms. Joachims [1998]

2.2.2 Linear Regression

Regression in the machine learning approach is used for prediction of a variable given a list of features and predetermined predictions. In simple terms we plot the given features with the given predetermined prediction and we find the best hyperplane which fits the data. This line is based on minimizing the least squares error. Formally it is defined as follows: we are given a training set which is defined as follows: $T = (x_1, y_1), (x_2, y_2), ..., (x_k, y_k)$ with size of k. And we have each x_i is a vector of size n and each y_i to be the predictions for the features x_i. We would like to find the regression coefficients β which best fit the data. In other words we would like to find $\vec{y}$ which is of the form:

$y = \mathbb{X}\beta + \epsilon$ where y is the dependant variable which we would like to predict, X is a matrix of x which are a list of features, β is the regression coefficient vector which is used for estimation and, ϵ which accounts for the noise. This can be solved using the Ordinal Least Squares method which minimizes the errors between the all data points. T. Tony Cai [2006] In our work, we use linear regression to predict the scores of sources using features such as percentage of art articles, traffic rank and other similar metrics.

2.2.3 RankSVM

We will be using pairwise ranking method for this as this is the one which is used mostly on web pages and ranking them. To start our goal is given a query q, and two instances of pages or articles to choose which one is more relevant. For these two pages the features can include website traffic measure, Page Rank, relevancy of page to article, and other related features which give an accurate description of the pages. RankSVM turns this problem into a SVM classification problem and it finds weights based on the features and labels which are given. To start we are given a list of features in a vector, $\vec{x_i}$ and we are given two instances of feature vectors for a query where q is constant. We can then turn this into an SVM classification problem as follows: We create two classes, relevant and not-relevant and we give training data with such labels $T_q = (x_1, l_1), (x_2, l_2), ..., (x_k, l_k)$ where the query is constant between each two feature vectors. Then this turns into a classification problem where each x_i is a list of features and the l_i is the label. We then use an SVM classifier to solve the optimization problem portrayed above and find the maximum separating hyperplane. This allows the use of many different features to predict relevancy of web pages and it can be catered to domain specific problems. Zhe Cao [2007]

Chapter 3

DATASET

3.1 Collection

We start with a database of well-known names given from a trusted source such as artfacts. We would like to find sources where these names are mentioned and use those sources as our primary base for news articles. To start with this task we have two options. One approach to this is the collection of news websites based on searching for them from search engines. Although, this is one way to do it, this approach however is not well suited for this task due to many different categories of art and also websites utilizing SEO would not give accurate results. Instead we start with names which we extract in Chapter 6. After extracting these names we start to find news websites which mentioned these names based on searching a search engine such as ask.com. Since there are thousands of names we can see that overall we are bound to get websites which are popular news websites and have a variety of artists mentioned. Thereafter, we find the top k domains which appear the most from these search engine results, in our case 150 and we use these domains for gathering news articles and exploring new artists based on their media ranking.

3.2 Data Set

3.2.1 News Aggregators

Our first idea was to use a news aggregation sources such as Moreover which are websites that gather, analyze and provide content from many different sources in one place. Usually the content from news aggregators is similar to an RSS from news websites with headlines, author, publishing date and category. The first and foremost problem with these types of services in our case is that they are from general news sources which include many different categories. They are for websites like Google News or Yahoo News for the general news. In our case we required articles only on Art related content and they were lacking. Secondly, the cost to use these services is unnecessary and can be easily substituted with RSS. Furthermore, its not very flexible as adding new sources would require time and money and and would not be able to add sources automatically.

3.2.2 Non-RSS

We have all these sources and we would like to have consistent news articles from them related to the topic. The second idea was to use a crawler like Nutch or Scrapy and crawl the website to a consistent depth, in our case, 2. We then consistently use it to get articles that appear on front page. However, with this approach there is a high risk to get blocked from websites and also it is very inconsistent because many websites have archives and other entities on their websites which we don't consider to be of use. Article detection was also difficult as websites had a variety of messages, announcements, auctions and not a clear guideline between content and non-content. Furthermore, many websites which didn't have much new and updates content on their website were being used which was not very helpful in calculating media ranking.

3.2.3 RSS

RSS (Rich Site Summary) is a web service that is used by millions of websites and blogs which allows them to publish frequently updated material such as blog entries and news headlines. An RSS also termed as a "feed" usually contains either a full text of the article or entry or short summary/text. Metadata such as tags and categories, publishing data and author's name are usually included as well. The main reason we would like to use this is so that the material we get is easily obtainable, customizable and it is frequently updated. The main way, it is used is by software applications called "RSS reader", "aggregator" or "feed reader". Generally RSS files are published in XML format and can be easily read by humans and can easily be parsed.

We then considered another idea which was to use RSS which is used to get frequently updated information from websites such as blog entries, news headlines. This approach helps in getting updated content so the ranking of the artist can be up to date and there was a clear distinction between updated content and non-content. Furthermore, the ranking of the artist can be updated based on recent mention in the media. We then started to make the list of websites which we wanted to crawl for RSS feeds. We used the method discussed in the Collections section of the current chapter. We then started to go to each website and crawl it for RSS feeds. One problem, however is that an RSS document primarily includes summarized text and metadata and doesn't include the full article, whereas if we follow the link to the HTML of the website , we see that it has many junk content such as ads, related content, comments. We then have the problem of extracting an article given the source HTML and a summary from the RSS.

3.3 Extraction

One obvious approach to extract text from a web page is to convert all non-tags on the page to text and use that as the content. For general purpose that would work fine, however in our case we have to match the article contents to names of artists. In many galleries we come across bad terms such as "Contact Us" and if we do not remove them from the article then many of these terms will appear on the top. To counter this, we then have the following goal: to take a web page, perform calculations and extract the main text of the article without side content. We use the algorithm which is presented in Chapter 2. The basic algorithm is as follows: We start with finding all the DOM Elements of the HTML page and we extract the blocks in HTML which contain the most text data. We then sort them by highest length to smallest and we compare the block to the full document so that we can compute the percentage of each blocks tokens extracted as content and use a threshold to determine its textual density. We then use another feature which is the text to tag ratio to find the main content. This approach is built into a library for Python which we use to find the main content.

Chapter 4

DATA MODEL

Now that we have crawled and parsed the data, the next reasonable step is to used the parsed data and create database tables.

4.1 Useful Data Extraction

We therefore initialized a scraper in the Python language known as Scrapy, using these we scraped all the links of the partial articles in the RSS feed which we have for each website. We then proceeded to extract the main content of the web page as well as meta data which includes link of the website, the title, the domain, the date and time and we store this in the master table. After continually inserting into the database for a few months, the table contained around 70,000 articles from only art related websites. This table is the key for the next steps in classification and scoring.

4.2 Creating Data Models

Our next step is to first define the database models so that it reflects the data we intend to store.

4.2.1 Defining Schema

We begin the define the schema of the most important table, “rss_content” which carries all the article data along with all the meta data.

- **link varchar(200)**: We use this key as the primary key as we want each article to be different and therefore we choose a unique link.

- **domain varchar(50)**: Since we have over 150 different sources, we would like to keep track each source and its many different attributes. Therefore, we use this field so that we can rank and analyze different sources as well as analyze artist coverage over the different news sources.

- **date datetime**: Given that we have collected news articles for artists for over 6 months, we would like to keep track how each source is faring over time and how different artists are covered in different sources at different times. Furthermore, the main reason we have this field is so that we can discover new unknown artists and see the time when they start to gain popularity to when they peak in popularity. Unlike music and other industries, we noted that they don't happen over a short amount of time, rather a period of few months.

- **title varchar(1000)**: By intuition we know that one of the best ways to find the relevance of an individual is mentioned in an article, is to first check the title. Since the title is the main relevance point of an article we use this to find relevance of an artist to a particular article.

- **content longtext**: This field of the table is the most important attribute because it carries the actual content of the article. Since cleaning the data is very important before it is stored, we have cleaned it using the algorithm discussed in Chapter 3.2. It discusses how we extract the article and how we store it before it is analyzed or before the classification task which is discussed in the next chapter.

Chapter 5

CLASSIFICATION

We weren't able to directly use extracted articles to find artist trends and calculate their media score because a number of the sources were not 100% clean. Many of the popular sources such as latimes.com and theguardian.com had music and entertainment mixed in the with art related news. Furthermore, many articles even from the art related domain still did not contain art related news but maybe news about the artist not related to art. For example there were articles on political stance of artists, death of artists, and about their personal life.

We explored many different options for classifying articles as to avoid non-related articles. We will mention all the approaches that were used and furthermore, analyze accuracy for each method. Our first option was to classify articles solely based on mention of known artists in the article. Our second option was to find names of artists for other categories such as Music, Movies, and News and classify artists based on those names. Our third option was to use text categorization with Support Vector Machines using different sources for each category. Our last and final option was to use text categorization with Support Vector Machines using hand picked articles for each category.

5.1 Classification by Names

Our first intuition was to classify the article based on the number of names that appear in the article. The main reason behind this intuition was that depending on the number

names which occur in an article that is an artist we can determine if the article is art related or not. However there are 3 main reasons for this approach not working.

1. Firstly, that the names we have collected which we discuss in chapter 6 are not unique to artists and therefore other entities in other categories also have these names.

2. Secondly, even though the names are mentioned in the article we see that these articles talk about political stance of artists, death of artists, and about their personal life which don't constitute as art.

3. Lastly, we saw that many articles talk about multiple aspects in terms of categories such as music and art and are not strictly art. Therefore these articles have mention of the artist but in a very small part of the article.

To counter the first and third point we looked to names of individuals in other categories to see if having positive and negative names would help in distinguishing art and non-art articles.

5.2 Classification by Names and Negative Names

Since classification by names of artists was not enough, we started to find other categories which were also included in our rss data. We saw that the most prominent categories which were inside the database alongside the art articles were music/entertainment, and movies. Using these observations, we started to look for databases with musicians and films. For the music database, we found that the "discogs" database was the most complete with around 5 million artists. We also used the most complete movies information which is provided by IMDB of movie actors, directors, and producers. We parsed and imported these large databases into our database and used

them for the purpose of classifying the articles. The problem with this apporach is that there are many articles with multiple names from each category and it is difficult to classify articles in this scenario because many artists and musician share the same names, therefore it is very difficult to distinguish the context in which the names are mentioned. Furthermore, many articles have multiple names, names from artists and musicians. In this case, if classification of the article is purely based on which category has more names than this can pose to be inaccurate. Suppose a case where there are two names of musicians matched and one artist. And suppose the musician names are common names. They can easily be names of artists which are unknown and based on this the article would be classified incorrectly.

The primary reason, however for not ultimately not using the name scheme for classification is because of "Artist Discovery". We plan to find new undiscovered artist names and find their ranking and coverage in the media. However if we classify articles differently based on names then many undiscovered artists would stay undiscovered. Since in Chapter 6, we introduce an algorithm for artist name extraction it is necessary now to find another technique that is independent of names. In the next section we use an algorithm which is independent of names and instead uses the content of the article for classifying it. As we will see, this is much more versatile and accurate and that is what we ended up using for the application.

5.3 Classification by SVM Automatically

Given that classifying by names of artists and negative names of musicians didn't work as we would like them to. Therefore, we used a fairly popular technique for classifying known as SVM. We introduced this in Chapter 2 and we tried multiple techniques which are presented as follows.

The first technique we tried was to gather articles from each source by rss. We had

around 150 sources for art, so we looked for rss for other categories which included news, music, and films. The plan was to use each articles from each category as defined by rss to use training data and use the resulting classifier to manually test articles and see how they fare. We wrote a web page which will test the article and return the classified result.

To start, we first selected all the articles from the database for each category. Then, we split each document from each category into words and we found the term frequencyinverse document frequency for each word in each category. We did this for all the 3 categories which we had which was art, music, and movies. We then used SVM with the bag-of words model using sparse features to build a classifying model. We saw that the tf-idf representation included many non-art related words in the frequencies due to many articles having non-art related terms. This lead to a high importance to non-art related terms and therefore it classified based on the erroneous frequencies. To counter this, we tried another method which we outlined in the next section.

Our next idea was to use "cleaned" articles which we defined as articles which have only artist names and no other names which are ambiguous. We define ambiguous names to be names which musicians and artists both have. We then removed all the articles with these names for all three categories and we did a similar technique with the last section and we train the SVM classifier based on these documents. We tested the classifier based on similar tests from non clean articles and we see the following results.

Art

art	1.040005335
2015	1.103184236
work	1.170625517
artist	1.194156014
new	1.218253566
artists	1.242946179
like	1.268263987
works	1.268263987
click	1.376477571
gallery	1.376477571
museum	1.376477571
enlarge	1.405465108
way	1.405465108
time	1.435318071
exhibition	1.435318071
years	1.497838428
street	1.530628251
painting	1.564529803
space	1.564529803
view	1.564529803

Music/Movies

like	1.194156014
time	1.218253566
new	1.218253566
music	1.242946179
just	1.268263987
years	1.376477571
song	1.376477571
songs	1.405465108
set	1.435318071
people	1.46608973
right	1.497838428
way	1.530628251
album	1.530628251
day	1.564529803
got	1.564529803
know	1.564529803
love	1.599621123
2015	1.599621123
don	1.599621123
old	1.635988767

News

said	1.081678031
year	1.435318071
2015	1.46608973
people	1.564529803
monday	1.599621123
country	1.635988767
years	1.635988767
government	1.673729095
told	1.673729095
share	1.673729095
new	1.712949808
news	1.712949808
make	1.753771802
old	1.796331417
president	1.840783179
day	1.840783179
according	1.936093359
week	1.936093359
related	1.936093359
right	1.987386654

Figure 5.1: TF*IDF of Automatically Classified Articles

As we see, it doesn't improve much compared to the non-clean articles. We hypothesize that this is due to the high volume of articles and the fact that many of the articles have words which although are not common, they become common over the corpus even though they don't carry much weight. For example here are a few terms which are ranking fairly high in their tf-idf representation, however are not art related. Another reason we observe is that, even though many articles have names, they are not related to art. For example we saw many articles about the political stance of artists, or death of artists, or a tribute to the artist. All of these have relation to the artist but not actual art. Since, we would like to use the application for artist discover, these types of articles are of no use and they improperly skew the training data to useless words. In the next section we choose out best performing technique which fixes these problem.

5.4 Classification by SVM Handpicked

Our next and final option was to handpick articles from each of these sources and use this as a basis for classification for all the articles. We started by going to famous art news websites such as hyperallergic.com, artnews.com and others and started to find articles which were related only to art independent of artists. We then did the same for music/entertainment articles as well as news articles. We then accumulated around 100 articles for each class and saved it to the database with manual labelling. We then proceeded to split the data to 50/50 for training and testing. We then trained the SVM classifier on the training data and proceeded to test it on the testing data. To our surprise we got around 99% accuracy. To test for any errors we mislabelled a few of the articles in the categories and proceeded to test them. We found out that the SVM classifier also correctly labelled them even though we gave an incorrect labeling. We saw the following results.

Art

art	1.040005335
2015	1.103184236
work	1.170625517
artist	1.194156014
new	1.218253566
artists	1.242946179
like	1.268263987
works	1.268263987
click	1.376477571
gallery	1.376477571
museum	1.376477571
enlarge	1.405465108
way	1.405465108
time	1.435318071
exhibition	1.435318071
years	1.497838428
street	1.530628251
painting	1.564529803
space	1.564529803
view	1.564529803

Music/Movies

like	1.194156014
time	1.218253566
new	1.218253566
music	1.242946179
just	1.268263987
years	1.376477571
song	1.376477571
songs	1.405465108
set	1.435318071
people	1.46608973
right	1.497838428
way	1.530628251
album	1.530628251
day	1.564529803
got	1.564529803
know	1.564529803
love	1.599621123
2015	1.599621123
don	1.599621123
old	1.635988767

News

said	1.081678031
year	1.435318071
2015	1.46608973
people	1.564529803
monday	1.599621123
country	1.635988767
years	1.635988767
government	1.673729095
told	1.673729095
share	1.673729095
new	1.712949808
news	1.712949808
make	1.753771802
old	1.796331417
president	1.840783179
day	1.840783179
according	1.936093359
week	1.936093359
related	1.936093359
right	1.987386654

Figure 5.2: TF*IDF of Handpicked Articles

As we can see the accuracy of classification for this model was very high in comparison to the model using artist names and using RSS classified training data. We checked the training data tf-idf scores and we saw that many of the words which were highly ranked were art related terms. This is due to human intuition choosing strictly art related articles rather than articles loosely related to art.

In the end we found the SVM based on the Handpicked articles works best as they were chosen from many different sources and were strictly art. Furthermore, they included many broad art related terms in conjunction with music and news articles which also included many strictly music and news related terms.

Chapter 6

ARTIST NAME EXTRACTION

6.1 Website Selection

Our first and most important step is to first identify websites which have the data which we can extract. The reason that it is most important is that it determines the quality of data we get and it also has an impact on the results we observe. Our criteria for finding websites to crawl for artist name extraction were the following:

1. Gallery Website
2. Contains A Large Collection of Artist Names
3. Artist Name is Extractable (Not embedded In Image)

Based on these factors we started to discover websites. Some of the websites which we crawled or used as a source of crawling include:

- www.artfacts.net and similar websites
- Google Places API to find extensive list of galleries
- Using search engine results to find more websites with names.

6.1.1 Large Dataset Websites

We are given large data set websites which have many different names which we classify as known. We would like to extract these names so that we can differentiate between

known and unknown artists in different galleries. For this extraction task, we use a scraping framework known as Scrapy which allows us to navigate website pages and extract specific defined content based on the XPATH representation of specific HTML tags. We use this to scrape around 200,000 artists from the known websites and store it in our database.

6.1.2 Gallery Websites

We see from many examples that the majority of websites in the art/gallery domain don't follow any specific structure and each website is built upon personal preference. Few examples represented show a part of different websites which we plan to crawl.

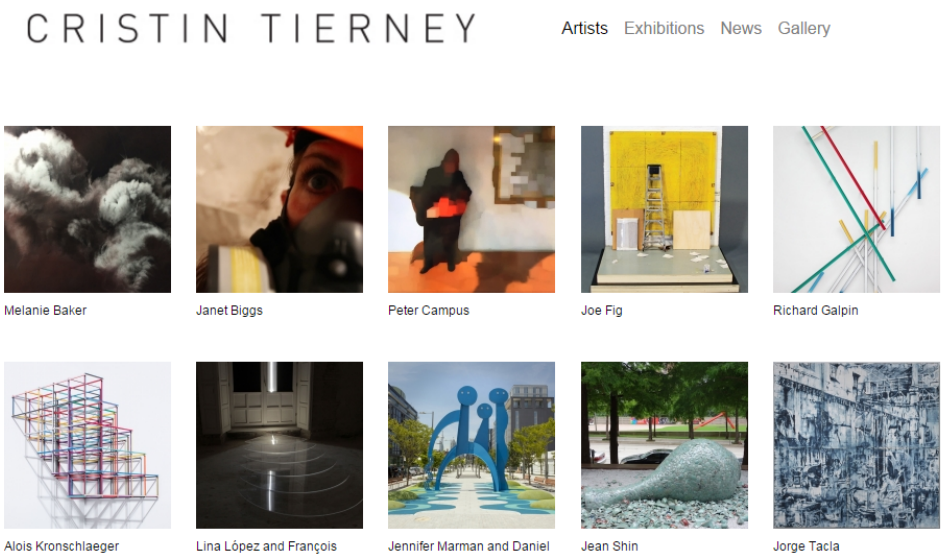


Figure 6.1: Example 1 of Website Structure

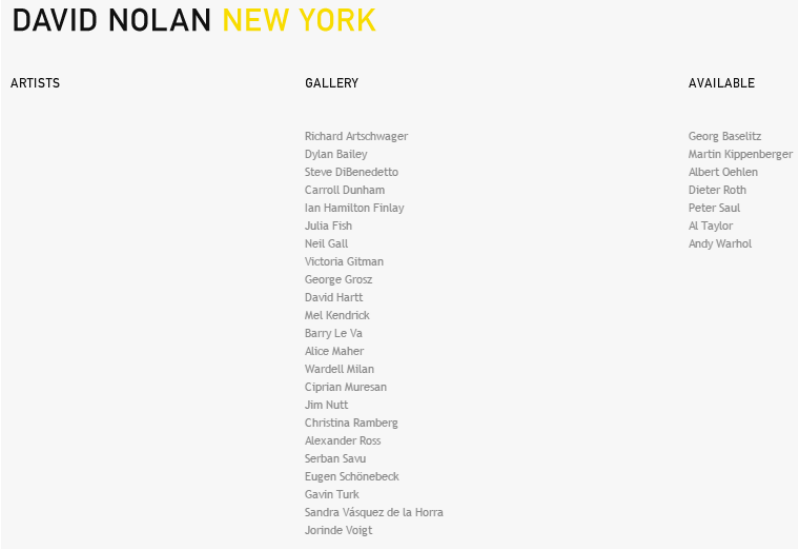


Figure 6.2: Example 2 of Website Structure

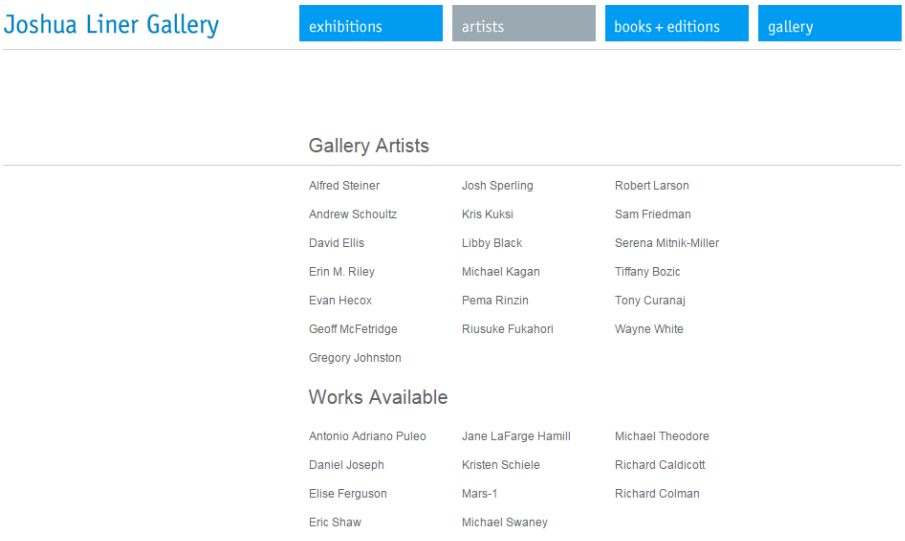


Figure 6.3: Example 3 of Website Structure

As we can see from the images, the websites all have different structures and have no specified layout. The challenge lies in generalizing extraction of names for these website using their HTML and keeping good accuracy which adapts to many different layouts and structures. The challenges include:

1. Generalizing Extraction for Gallery Websites

2. Extracting Names with High Accuracy and Consistency

3. Avoiding Terms which might look like Names but are instead "Junk"

Having these points in mind, we started to analyze the HTML of all these similar websites. We noticed a trend that the websites follow which we outline in the following sample image.

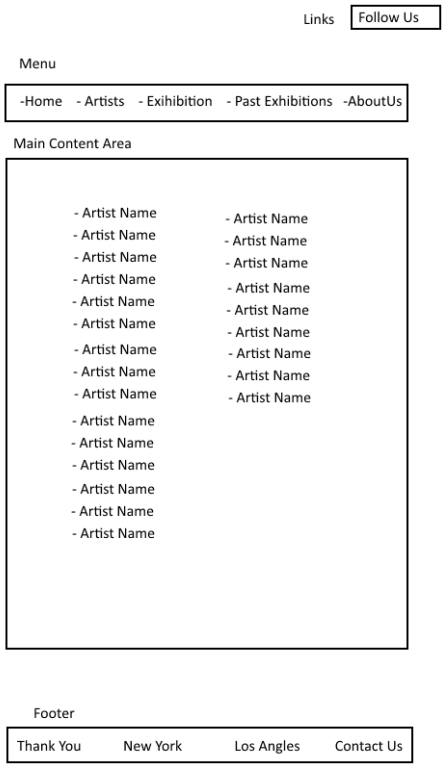


Figure 6.4: General Outline of Gallery Website

We see that artist names are usually in a list of some sort, either individually or with and image and they are the most dense area in the website page. Furthermore, we see however that many terms that look like artist names, but however are not such as:

About Us, Contact Us, Past Exhibitions, New York etc. are included in such HTML documents. To make an algorithm for this we have to first parse the HTML, iterate the DOM and manipulate the DOM using a query language called XPATH and extract terms and analyze them.

We present an algorithm which eliminates them. Related Algorithms in extracting names use linguistic analysis and natural language processing, however this will not be useful in this case as they are presented in a different fashion in the art related domain.

6.2 Extraction

6.2.1 Algorithm

We start by fetching the HTML document from the gallery website, and we filter the artist URL based on many different regular expressions based on our observations. We then use the following simplified regular expression to filter terms which look like names in the HTML document:

(?<=>)[a–zA–Z]+ [a–zA–Z]+(?=<)

In simple terms, it is a two worded term which is enclosed in HTML tags (Names with more than two words or other alphabets can be added to this as well without much modification). We then run the following algorithm to get rid of the "Bad terms" which are present in the HTML documents which can skew the results for artist score.

We start by first writing an XPATH query which iterates and finds HTML elements based on the query. We start by finding the Least Common Ancestor HTML Tag of all pairs of these terms and store it. We then iterate through all these starting from

HTML tags which have the least amount of terms that is greater than 1. We then use the following heuristic based on our observations:

1. If the HTML element has less than k in our case we set this to the average number of elements in a menu which look like names. In our case we calculated it to be 3 terms. If we see this HTML block then we guess that it is a menu with high probability and not a list of artist names as we see that they come in lists which have many of them bunched together.

2. Since we are iterating from smallest number of terms to largest, if we see an HTML Element with x terms and then next HTML Element has x+1 or x+2 terms we see that they are outlying junk terms for example: Follow Us, About Us. etc.

3. Thirdly, we use common art related terms extracted from art related articles. We find terms sorted based on their tf-idf score and choose the top-k. We then eliminate terms or HTML Elements filled with terms based on a threshold for occurrences.

4. Lastly, we use an element of learning to detect junk terms. Every time we see a junk term in a gallery we add it do a database. We see that as many of these terms accumulate over time the likelihood that the junk terms appear again and again are really high. We then use this to eliminate terms or HTML Elements filled with terms based on a threshold for occurrences.

6.2.2 Example

Here we present an example visually of how the algorithm works. We are given the following HTML Document:

Listing 6.1: Example of General Gallery Website HTML

```
<html>
   <body>
      <div class="header">
         <ul>
            <li> Home</li>
            <li> Artists</li>
            <li> Exhibition</li>
            <li> Past Exhibitions</li>
            <li> Current Exhibition</li>
            <li> Upcoming Exhibition</li>
         </ul>
      </div>
      <div class="content">
         <ul>
            <li>Coy Cole</li>
            <li>Neil Hilger</li>
            <li>Armando Melin</li>
            <li>Jordan Jeppson</li>
            <li>Phillip Neace</li>
            <li>Ahmed Merriweather</li>
            <li>Nathan Eshbaugh</li>
            <li>Myron Paterno</li>
            <li>Clifton Devine</li>
            <li>Jarrett Lyall</li>
            <li>Keven Yount</li>
            <li>Cortez Isherwood</li>
            <li>Hank Cusick</li>
            <li>Stefan Glynn</li>
            <li>Bert Aylesworth</li>
            <li>Johnathon Mcintyre</li>
            <li>Laurence Mcgowin</li>
            <li>Minh Hornsby</li>
            <li>Denny Tacker</li>
            <li>Maria You</li>
         </ul>
         <div>Follow Us</div>
      </div>
      <div class="footer">
         <ul>
            <li> Contact Us</li>
            <li> About Us</li>
```

```
        </ul>
      </div>
   </body>
</html>
```

We then filter out all terms based on the regular expression mentioned above, which in this case is all the terms. We then find the LCA between all pairs of terms which gives the following HTML Blocks sorted from smallest to largest:

Listing 6.2: Block 1 of HTML Extraction

```
<ul>
  <li> Contact Us</li>
  <li> About Us</li>
</ul>
```

Listing 6.3: Block 2 of HTML Extraction

```
<ul>
  <li> Home</li>
  <li> Artists</li>
  <li> Exhibition</li>
  <li> Past Exhibitions</li>
  <li> Current Exhibition</li>
  <li> Upcoming Exhibition</li>
</ul>
```

Listing 6.4: Block 3 of HTML Extraction

```
<ul>
  <li>Coy Cole</li>
  <li>Neil Hilger</li>
  <li>Armando Melin</li>
  <li>Jordan Jeppson</li>
  <li>Phillip Neace</li>
  <li>Ahmed Merriweather</li>
  <li>Nathan Eshbaugh</li>
  <li>Myron Paterno</li>
  <li>Clifton Devine</li>
  <li>Jarrett Lyall</li>
  <li>Keven Yount</li>
  <li>Cortez Isherwood</li>
```

```
            <li>Hank Cusick</li>
            <li>Stefan Glynn</li>
            <li>Bert Aylesworth</li>
            <li>Johnathon Mcintyre</li>
            <li>Laurence Mcgowin</li>
            <li>Minh Hornsby</li>
            <li>Denny Tacker</li>
            <li>Maria You</li>
          </ul>
```

Listing 6.5: Block 4 of HTML Extraction

```
          <div class="content">
            <ul>
              <li>Coy Cole</li>
              <li>Neil Hilger</li>
              <li>Armando Melin</li>
              <li>Jordan Jeppson</li>
              <li>Phillip Neace</li>
              <li>Ahmed Merriweather</li>
              <li>Nathan Eshbaugh</li>
              <li>Myron Paterno</li>
              <li>Clifton Devine</li>
              <li>Jarrett Lyall</li>
              <li>Keven Yount</li>
              <li>Cortez Isherwood</li>
              <li>Hank Cusick</li>
              <li>Stefan Glynn</li>
              <li>Bert Aylesworth</li>
              <li>Johnathon Mcintyre</li>
              <li>Laurence Mcgowin</li>
              <li>Minh Hornsby</li>
              <li>Denny Tacker</li>
              <li>Maria You</li>
            </ul>
            <div>Follow Us</div>
          </div>
```

Listing 6.6: Block 5 of HTML Extraction

```
          <body>
            <div class="header">
              <ul>
                <li> Home</li>
                <li> Artists</li>
                <li> Exhibition</li>
                <li> Past Exhibitions</li>
```

```
                <li> Current Exhibition</li>
                <li> Upcoming Exhibition</li>
              </ul>
            </div>
            <div class="content">
              <ul>
                <li>Coy Cole</li>
                <li>Neil Hilger</li>
                <li>Armando Melin</li>
                <li>Jordan Jeppson</li>
                <li>Phillip Neace</li>
                <li>Ahmed Merriweather</li>
                <li>Nathan Eshbaugh</li>
                <li>Myron Paterno</li>
                <li>Clifton Devine</li>
                <li>Jarrett Lyall</li>
                <li>Keven Yount</li>
                <li>Cortez Isherwood</li>
                <li>Hank Cusick</li>
                <li>Stefan Glynn</li>
                <li>Bert Aylesworth</li>
                <li>Johnathon Mcintyre</li>
                <li>Laurence Mcgowin</li>
                <li>Minh Hornsby</li>
                <li>Denny Tacker</li>
                <li>Maria You</li>
              </ul>
                    <div>Follow Us</div>
            </div>
            <div class="footer">
              <ul>
                <li> Contact Us</li>
                <li> About Us</li>
              </ul>
            </div>
          </body>
```

We then start with the smallest block as see that it has number of terms to be 2 so we add the terms to the list of bad terms based on condition 1 of the algorithm. We then go to block two and we see that based on condition 3 and 4 of the algorithm that these are common domain related terms and we have seen them in the past. So we further classify this bad terms. We then go to block three and we see that it satisfies

all conditions and we add it to the list of good terms. In Block 4 we see the new term "Follow Us" and we see that based on condition 2 of the algorithm that it is an outlying term and we classify it as bad. In Block 5 there is nothing new so we continue and end the algorithm.

6.2.3 Analysis

To start the accuracy analysis, we start with a sample of 38 random galleries chosen from Chelsea, Manhattan which have a dedicated artist page. We then started to run our algorithm on these galleries and see the percentage of extracted terms with the total number of terms on the website, the missed terms which the algorithm didn't extract as well as the terms which are classified as bad which the algorithm extracted. In Table 1 we can see the data we extracted:

We normalized the counts and we can see in the graph Figure 1 the results. As we can see the majority of the names from the websites were correctly extracted.

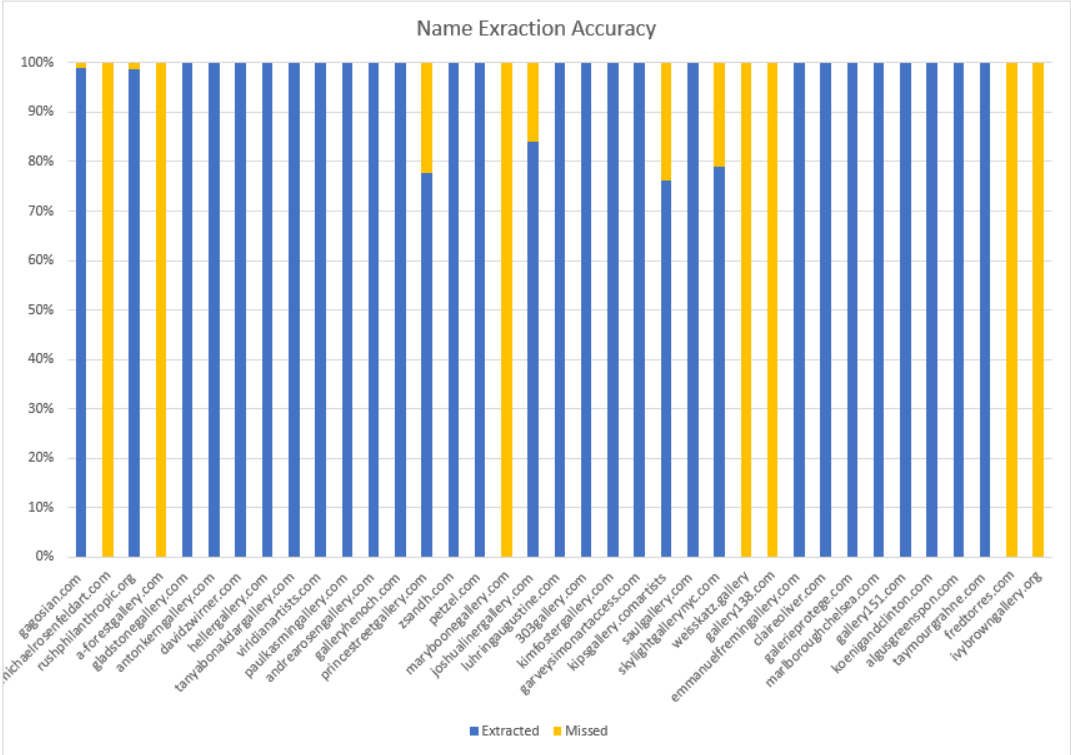


Figure 6.5: Names Accuracy

In fact, we can see that the algorithm is accurate for extracting names from the majority of website. However we can see that there are many websites which dont give any results for extracted names. We look at few cases below:

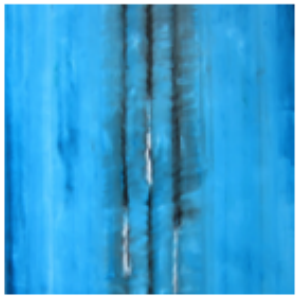
IVY BROWN GALLERY

675 Hudson Street, 4th floor, New York City, NY 10014
212.925.1111 www.ivybrowngallery.org

Artists

Cody S. Brothers
Photography

Elizabeth Gregory-Gruen
Sculpture - Cut Work



Todd Williamson
Paintings

Zoobs
Photography

Kenjiro Kitade
Sculpture

Home Contact Us Links facebook

Figure 6.6: Example 1 of Gallery which is difficult to extract

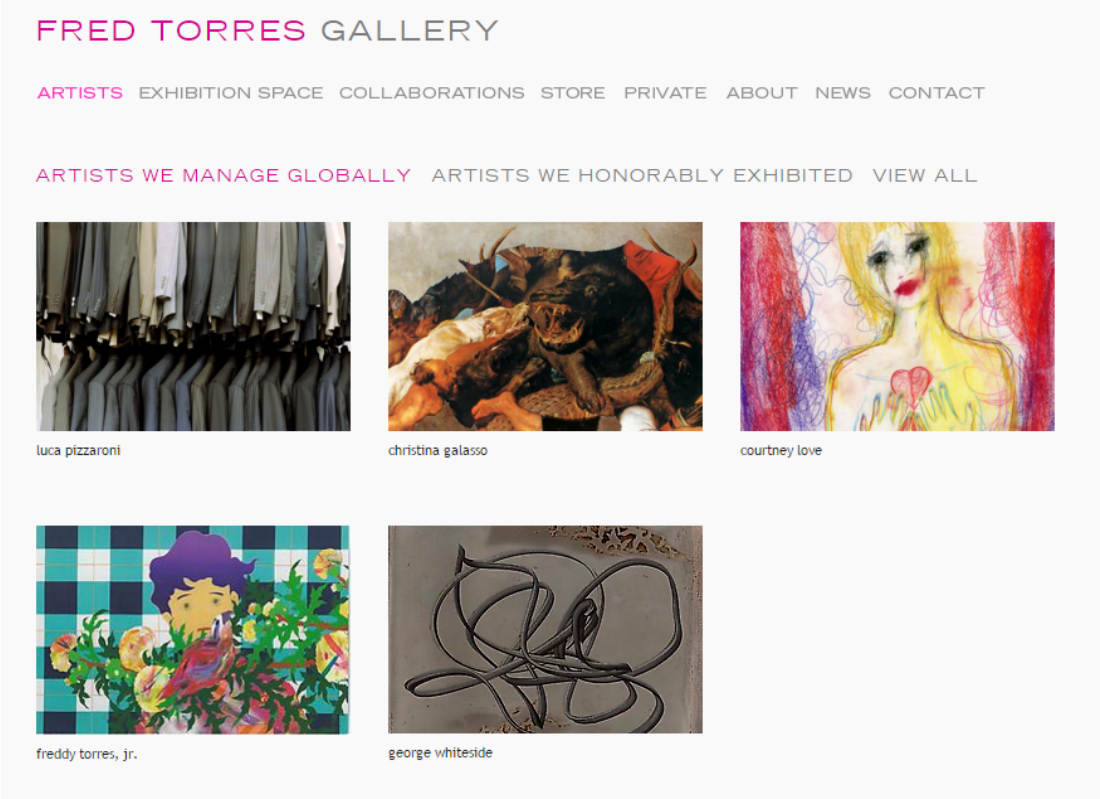


Figure 6.7: Example 2 of Gallery which is difficult to extract

We can see above that the names in the galleries are very sparse and there is not much density in the names. Another example is galleries which have text and names embedded in such a way that it is difficult to extract names with a simple regular expression and will require a much longer one.

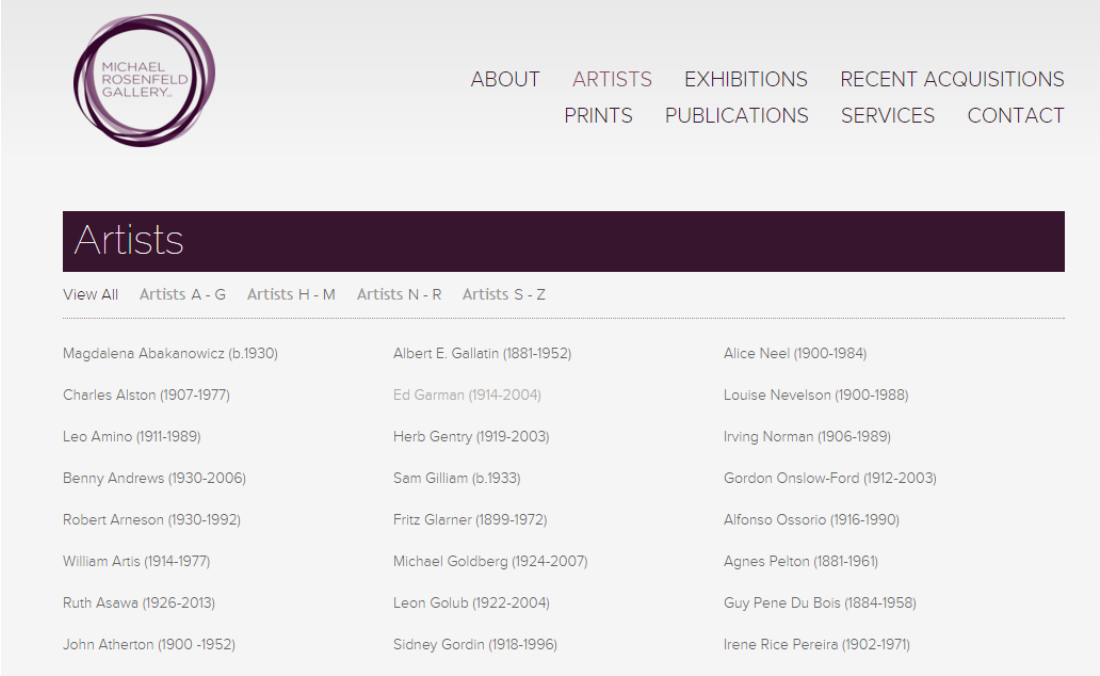


Figure 6.8: Example 3 of Gallery which is difficult to extract

Figure 6.9: Example 4 of Gallery which is difficult to extract

Lastly, many websites have artist names embedded in images. This makes it difficult

to extract names because they dont match the regular expression.

Figure 6.10: Example 5 of Gallery which is difficult to extract

Figure 6.11: Example 6 of Gallery which is difficult to extract

Table 6.1: Name Extraction

Website	Extracted	Bad Terms	Total
gagosian.com	107	0	108
michaelrosenfeldart.com	0	0	100
rushphilanthropic.org	79	1	80
a-forestgallery.com	0	0	51
gladstonegallery.com	49	0	49
antonkerngallery.com	49	0	49
davidzwirner.com	43	0	43
hellergallery.com	37	0	37
tanyabonakdargallery.com	34	0	34
viridianartists.com	33	0	33
paulkasmingallery.com	32	0	32
andrearosengallery.com	29	0	29
galleryhenoch.com	28	0	28
princestreetgallery.com	21	0	27
zsandh.com	27	0	27
petzel.com	27	0	27
maryboonegallery.com	0	0	26
joshualinergallery.com	21	0	25
luhringaugustine.com	24	0	24
303gallery.com	24	0	24
kimfostergallery.com	24	0	24
garveysimonartaccess.com	23	0	23
kipsgallery.comartists	16	0	21
saulgallery.com	20	0	20
skylightgallerynyc.com	15	0	19
weisskatz.gallery	0	0	17
gallery138.com	0	0	14
emmanuelfremingallery.com	13	0	13
claireoliver.com	13	0	13
galerieprotege.com	13	0	13
marlboroughchelsea.com	12	0	12
gallery151.com	11	0	11
koenigandclinton.com	10	0	10
algusgreenspon.com	7	0	7
taymourgrahne.com	7	0	7
fredtorres.com	0	0	5
ivybrowngallery.org	0	0	3

Chapter 7

MEDIA RANKING

Now that we have extracted the names and articles and have properly classified them, out next goal is given an unknown artist and a list of articles relating to that artist, we would like to determine their ranking. For this we have to take into account many different factors such as:

- The reputation of the source of each of the articles.
- Relatedness of the sources to our topic which is art
- Article relevance and dedication to the artist.

Based on these very important requirements we started to build the score for each of them. We looked and tested multiple approaches for each of these requirements.

7.1 Source Ranking

7.1.1 Traffic Based

Our first method to calculate the reputation for each source was to base it on the properties of the source which included: Traffic Signal, Percentage of Art Articles, and Number of Articles. Intuitively, this would make sense for ranking sources. One popular metric used on the web to estimate website traffic is: Alexa Rank. Since traffic is

an important measure for website reputation we assumed that this would make sense. We used this in conjunction with other features which we calculated such as number of articles collected from the source, and percentage of articles related to art from all the articles based on classification by SVM as we discussed in Chapter 5. Given that we had these features our next task was to rank the sources based on these signals. We took the Machine Learning approach and gave some training data of the final ranking of some of the sources based on common knowledge of art sources. We then used the method of linear regression introduced in chapter 2, to find the hyperplane that best represented the weights for the features we provided as input. We then got the following data:

What we observed is the following: Many of the famous news publications were ranked on the top such as nytimes.com, theguardian.com, npr.com. Since an artists coverage in these news sources mean that the artist has a high reputation either for a long time or recent reputation. However, we encountered two major issues with this approach. Firstly, we noticed that websites such as hyperallergic.com and whitney.com didn't rank as high. Given that these are niche art magazines and have a high reputation in the art related domain, they should have been given a higher ranking. The second issue we encountered was that although famous news publications have a higher traffic than most of the niche magazines, their traffic is mostly for news and business rather than for art. Therefore, their traffic ranking is not accurate and will be skewed towards non-art related content. Therefore we needed to find a more accurate way to find the reputation of each source.

7.1.2 Reference Based

To fix the issue of ranking not based on traffic but based on quality of art coverage, we look to a more concrete feature which takes this into account. Since one of the most important aspect of the reputation of a source is based on how often it is referenced, we choose Wikipedia as our source by which we base our reference on. Since Wikipedia is very strict with users choosing their references and would not let unknown sources such as blogs to be used as evidence for artist biographies and art related news. Therefore we can say that the reputation of an article can be based on its number of references over all of Wikipedia. This however is not enough because the references can be based on non art related articles. To fix this we filter for articles in Wikipedia which are art related and the source appears as a reference. Doing this we found some of the results as follows in Table 7.1.

As we can see, there are many big news publications on the top, however there are also many niche art news magazines and websites such as brooklynrail.com, complex.com and related websites. As we can see this is a good measure of trust for a given source because it is based on real world references to the source and is based on reliable editing from Wikipedia editors. Since it grows exponentially with the number of references for sources, we set the score of the source to $log(number\_of\_references)$

7.2 Article Relevancy

Now that we have the domain score, we wish to find out the relevancy of the artist for all the articles s/he are mentioned in. The next goal is to go through all the articles which were crawled from a given artist and rank the relevancy of the article. To do this we first started to observe the relevancy of the articles manually for a random set of artists. We found intuitively and based on our observations that given the contents of

the article which include the title and the body, if the artist is mentioned in the title then it is most certainly relevant to the artist. This is based on the observation that if the article contents are classified as art and if the artist is mentioned then we say that with a high probability it is relevant to the artist and it gets a really high relevancy score. If however the artist is not mentioned in the title then we calculate the article relevancy based on the number of times the artist appears in the article. Therefore, based on these conditions we find the relevancy of the article for each artist.

7.3 Artist Score

Since, we have found the two main components to each article in the database which is domain score based on references and articles relevance score based on the relevancy to the article, our final task is to combine these metrics and define a new score for each artist based on this. We start with finding all the articles related to a particular artist and we calculate the domain score and article relevancy for each article. Then based on all the scores from all the articles we combine it. However, the issue is that high ranking artists might have a low score because they are mentioned in many articles with little relevance. For example, the famous artist, Andy Warhol has thousands of articles while other unknown artists don't have many articles related to them. However Andy Warhol has hundreds of articles which have a low article relevancy score and unknown artist might have few articles but with high relevance. To counter this we add a factor to the equation which is proportional to the number of articles an artist has and therefore the popularity of the artist is included in the score. We find the score to be:

$$artist\_score = avg(source\_rank * (relevancy\_of\_article\_to\_artist) * (recency\_of\_article))$$

Table 7.1: Ranking of Domains based on References

domain	wikipedia art references
nytimes.com	15000
blogspot.com	11700
wordpress.com	7070
latimes.com	5930
telegraph.co.uk	5780
theguardian.com	4480
dailymail.co.uk	3550
huffingtonpost.com	3410
npr.org	2650
tumblr.com	1870
artnet.com	1460
nymag.com	1210
newyorker.com	850
typepad.com	838
complex.com	721
vulture.com	707
moma.org	657
vice.com	474
laweekly.com	456
artinfo.com	294
artforum.com	205
frieze.com	189
whitney.org	169
art-magazin.de	161
popsugar.com	145
brooklynrail.org	123
e-flux.com	122
artsjournal.com	121

Chapter 8

FUTURE WORK AND CONCLUSION

8.1 Similar Domains

The application we made which reflects the topics in thesis is related to the domain of art. Since these algorithms are domain specific however general enough to be applied to other domains easily. One case which we delve into is Restaurant Menu Items. Since many restaurant websites have a similar structure to art gallery websites where they have a page dedicated to menu, we can crawl menu items from these websites as our "names". We can then crawl aggregate and extract articles from food related websites in a similar fashion to how we did for art related websites. We can then use articles and calculate scores for food. Therefore we can do a similar idea where given an area, the application crawls restaurants in the area and find the "hottest" foods in the area based on ranking from food related websites.

8.2 Extensibility

Since the application that is built is fairly basic it can easily be extended to include a number of features. One feature which can easily be added is the ability to automatically detect sources form articles. We can find links in the current articles and see related domains and we can automatically add them. Over time the articles for the sources will accumulate and we can either keep the source or remove it based on few signals like art related content percentage and if any of the artists appear on any of the articles from

that source.

8.3 Conclusion

Art Ticker can be helpful to a collector or art agent in identifying new emerging artists. It can also be used by artists to see how they stand in comparison to other artists. To our knowledge it is first such system which can dynamically and automatically assist in the process of artist discovery.

Art Ticker is based on two phase technique: first we extract names of artists from thousands of galleries around the world, then we measure the media presence score of these artists in the news feed which we generate from hundreds of art news sources. These news sources include major media outlets as well as small, niche art publications. To this aim we develop the extraction methodology which benefits from homogeneous structure of gallery web sites. In order to clean the art news feed we utlize support vector machines (SVM) to distinguish visual arts publications (what we are interested in) from the music and movies articles which are often found in news sources that we utlize.

1. Objects (in our case artists) are presented in possibly very large number of similarly structured web sites. Thus, information is not aggregated

2. Media presence is an indication of emergence, of a positive trend

For example, stand up comedians often perform in small clubs, before making it big (small chance!), theater actors, musicians performing in clubs, public lectures (listed at small venues like libraries) with lecturers as entities to be discovered, finally professors who are listed on sites of academic departments (who generates most buzz?), Products (gadgets, books, movies) usually appear on large aggregator sites such as Amazon,

Netflix, IMDB etc are are less likely candidates of our two phase process (Extract first, then rank for media presence).

The question might arise: Is media score a good predictor of emerging and "making it"? This is an open question which requires much longer data collection and data validation and is a good foundation for a future study.

Bibliography

Wolfgang Nejdl Christian Kohlschtter, Peter Fankhauser. Boilerplate detection using shallow text features. 2010.

Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. 1998.

Matthew E. Peters and Dan Lecocq. Content extraction using diverse feature sets. 2013.

Juan Ramos. Using tf-idf to determine word relevance in document queries. 2003.

Jakob Busk Srensen. Support vector machines for pixel classification. 2014.

Peter Hall T. Tony Cai. Prediction in functional linear regression. 2006.

Jiawei Han Tim Weninger, William H. Hsu. Cetr - content extraction via tag ratios. 2010.

Anastasios P. Ventouris. A news aggregator using semantic and data mining technologies. 2014.

Xijin Tang Wen Zhang, Taketoshi Yoshida. Tfidf, lsi and multi-word in information retrieval and text categorization. 2008.

Tie-Yan Liu Ming-Feng Tsai Hang Li Zhe Cao, Tao Qin. Learning to rank: From pairwise approach to listwise approach. 2007.