PREDICTING MUSIC GENRE PREFERENCES BASED ON ONLINE COMMENTS

A Thesis

Presented To

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Andrew Sinclair

June 2014

COMMITTEE MEMBERSHIP

TITLE:	Predicting Music Genre Preferences Based On Online Comments
AUTHOR:	Andrew Sinclair
DATE SUBMITTED:	June 2014
COMMITTEE CHAIR:	Foaad Khosmood, Ph.D. Assistant Professor of Computer Science
COMMITTEE MEMBER:	Franz Kurfess, Ph.D. Assistant Professor of Computer Science
COMMITTEE MEMBER:	Zachary Peterson, Ph.D. Assistant Professor of Computer Science

Abstract

Communication Accommodation Theory (CAT) states that individuals adapt to each other's communicative behaviors. This adaptation is called "convergence." In this work we explore the convergence of writing styles of users of the online music distribution platform SoundCloud.com. In order to evaluate our system we created a corpus of over 38,000 comments retrieved from SoundCloud in April 2014. The corpus represents comments from 8 distinct musical genres: Classical, Electronic, Hip Hop, Jazz, Country, Metal, Folk, and World. Our corpus contains: short comments, frequent misspellings, little sentence structure, hashtags, emoticons, and URLs. We adapt techniques used by researchers analyzing other short web-text corpora in order to deal with these problems. We use a supervised machine learning approach to classify the genre of comments in our corpus. We examine the effects of different feature sets and supervised machine learning algorithms on classification accuracy. In total we ran 180 experiments in which we varied: number of genres, feature set composition, and machine learning algorithm. In experiments with all 8 genres we achieve up to 40% accuracy using either a Naive Bayes classifier or C4.5 based classifier with a feature set consisting of 1262 token unigrams and bigrams. This represents a 3 time improvement over chance levels.

TABLE OF CONTENTS

LIST OF TABLES viii

LIST OF FIGURES ix

1 Introduction 1

2 Background 2

2.1 Machine Learning . . . 2

2.1.1 Classifiers . . . 3

Naïve Bayes . . . 3

C4.5 . . . 3

Support Vector Machines . . . 4

IB1 . . . 4

2.1.2 Performance Metrics . . . 5

Area Under ROC Curve (AUC) . . . 6

2.2 Stylistics . . . 6

2.3 Music Genres . . . 7

2.4 Communication Accommodation Theory . . . 7

2.5 SoundCloud Corpus . . . 8

2.6 Tools . . . 9

2.6.1 Waikato Environment for Knowledge Analysis . . . 9

2.6.2 Natural Language Toolkit . . . 10

3 Related Work **11**

3.1 No Country for Old Members: User Lifecycle and Linguistic Change in Online Communities . . . 11

3.2 Distinguishing Venues by Writing Styles . . . 12

3.3 Twitter Part-Of-Speech Tagging for All: Overcoming Sparse and Noisy Data 12

4 Methodology **14**

4.1 Text Processing . . . 14

4.1.1 Tokenization and Token Substitution . . . 14

4.1.2 Part-of-Speech Tagging . . . 16

4.1.3 Stemming . . . 16

4.1.4 Term Frequency - Inverse Genre Frequency . . . 16

4.1.5 Feature Extraction and Feature Families . . . 17

Token N-Grams . . . 17

Counts . . . 17

Dictionaries . . . 18

Part-of-Speech N-Grams . . . 18

5 Experimental Setup **19**

6 Results **21**

6.1 Feature Set and Classifier Combinations . . . 21

6.2 Varying the Number of Genres . . . 22

6.2.1 Statistical Significance Tests . . . 22

2 Genres . . . 22

4 Genres . . . 24

6.3 Per-Genre AUC Results . . . 24

7 Recommendations for Future Work **26**

8 Contributions **27**

9 Conclusion **28**

Bibliography **28**

A Full Experimental Results **33**

LIST OF TABLES

6.1 Per Genre ROC results using best feature set and classifier. 25

A.1 Accuracy (%) for Two Genre Experiments using Token Unigrams and Bigrams. 33

A.2 F-Measure for Two Genre Experiments using Token Unigrams and Bigrams. 34

A.3 Accuracy (%) for Four Genre Experiments using Token Unigrams and Bigrams. 35

A.4 F-Measure for Four Genre Experiments using Token Unigrams and Bigrams. 36

A.5 Accuracy (%) for each Feature Set with Eight Genres 36

A.6 F-Measure for each Feature Set with Eight Genres 36

LIST OF FIGURES

2.1 Supervised Machine Learning Pipeline 4

4.1 Text Processing Pipeline 15

6.1 Results for 8 Genre Experiments 21

6.2 Results for 4 Genre Experiments 23

6.3 Results for 2 Genre Experiments 23

Chapter 1

Introduction

Many online communities exist for users to share and discuss content. SoundCloud.com is one of these websites. It is dedicated to the sharing and discussion of music. Through SoundCloud's interface users are able to leave comments on a per song basis. SoundCloud categorizes the music uploaded to its servers as one of several genres. In this work we explore the link between comment writing style and musical genre. We use a supervised machine learning approach with several classes of features in order to predict the genre of music each comment is discussing. We go on to use our classification system to determine if we can accurately predict the genre of a song based on the classification of the comments made about it.

Corpora used by Natural Language Processing (NLP) researchers are mostly works of literature or prose, generally free of misspellings and consist of well-structured grammatical and coherent sentences. This is not the case for the vast majority of SoundCloud comments. They often contain misspellings; both typos and stylizations. They lack much of the basic structure found in literary text. They are rarely longer than a sentence and are often single words or phrases. They also contain constructs mostly seen in Internet text: emoticons, hashtags, urls, and responses ("@" followed by a username). Our literature research indicated no other analysis done on a corpus of SoundCloud comments. This led us to use techniques from the vast amount of work completed on Twitter-based corpora.

Chapter 2

Background

2.1 Machine Learning

Machine learning is a "computational methods using experience to improve performance or to make accurate predictions"[1]. Algorithms analyze past sample data in order to generate models for predicting characteristics of data presented to these algorithms in future. The building of these models is called training. Our method takes the supervised learning approach. In this approach all of the training examples shown to the learning algorithm are tagged with labels representing classes of data. Large sets of labeled training examples can be difficult to obtain since they often require that humans manually label each example. The source of our corpus made this task relatively easy. Supervised learning is generally used for classification and ranking problems[1].

Features are sets of attributes that are associated with an example. They are often represented as vectors. As an example, the features of a book may include: number of chapters, presence of certain words, weight (in the case of a physical book), and the literary period in which it was written. Examples are grouped into samples. Training samples are groups of examples and their labels which a learning algorithm uses to train. Validation samples are used "to tune parameters of learning algorithms." Test samples are "used to evaluate the performance of a learning algorithm." [1] Test samples are kept separate from training and validation samples and are used after a learning algorithm has been trained.

2.1.1 Classifiers

Naïve Bayes

A Naïve Bayes classifier is a classifier based on Bayes' theorem. It has a strong (naïve) independence assumption about features. It assumes the presence of a feature is independent of the presence or absence of any other feature. A Naïve Bayes classifier chooses the class label based on the class that is most probable. This means that a Naïve Bayes classifier must compute the probability that an instance is of each class. The probability of a class is the product of the conditional probabilities of each feature with respect to the class.

The equation is as follows:

$$classify(f_1, ..., f_n) = max(C = c) \prod_{i=1}^{n} p(F_i = f_i | C = c) \tag{2.1}$$

Where f_i is a feature, c is a class label, and i is the number of features.

C4.5

The C4.5 learning algorithm is a decision tree algorithm. It is an improvement on the ID3 algorithm. Both created by J. Ross Quinlan [2]. In the decision tree *leaves* indicate a class. *Decision nodes* specify some test to be carried out on a single attribute/feature value with a branch for each outcome. At each decision node the C4.5 algorithm uses the attribute which creates the greatest split of classes into subsets. The process recurs until C4.5 reaches one of three base cases.

- All training samples remaining are from the same class.
- None of the remaining features provide any information gain.
- The algorithm encounters and unseen class.

We make use of the WEKA J48 implementation of this classifier.

Support Vector Machines

Support Vector Machines (SVM) map training vectors, x_i into a higher dimensional space by a function ϕ. The mapping into higher dimensional space is defined in terms of a kernel function. There are many different kernel functions. Some of the most common are: linear, polynomial, radial basis function, and sigmoid. SVM finds a linear separating hyperplane with the maximal margin in the higher dimensional space [3]. The goal is to maximize the distance from the hyperplane to the nearest training data point. We make use of the WEKA interface to the LibSVM [4] implementation of SVM.

IB1

The IB1 algorithm is a very simple learning algorithm. It takes an "instance based" approach to predict the class of a sample. It finds the training instance with the smallest euclidean distance from the test example and predicts the training instance's class for the test instance. [5]

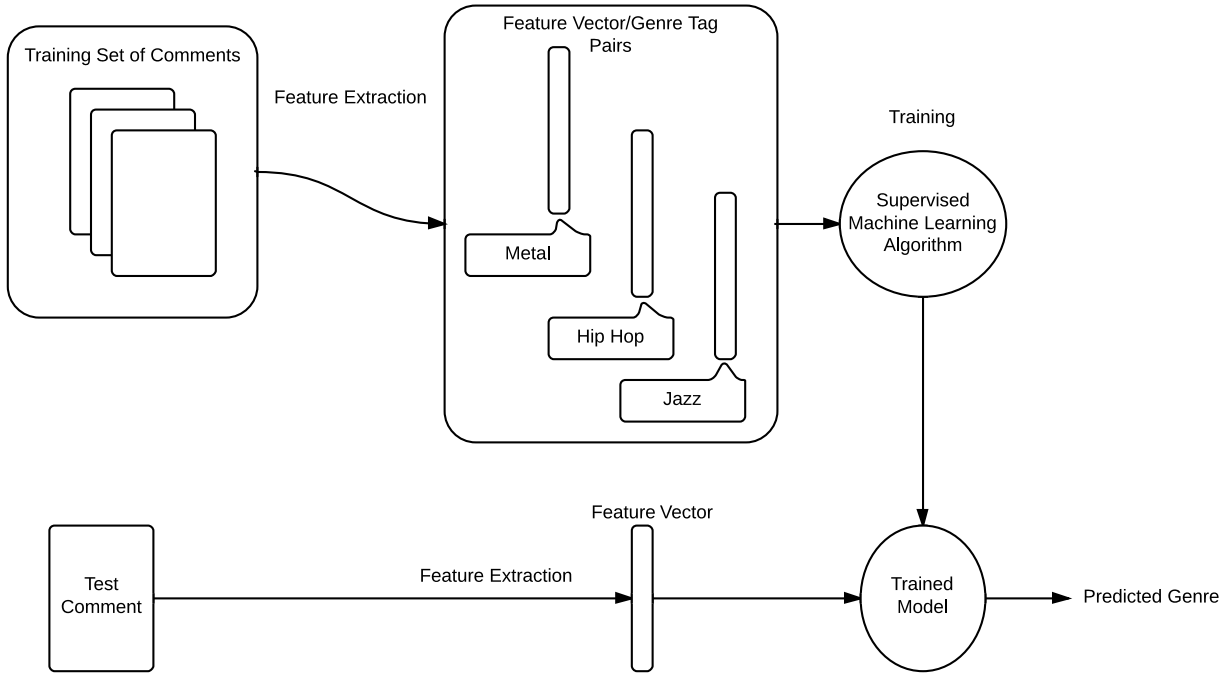

Figure 2.1: Supervised Machine Learning Pipeline

2.1.2 Performance Metrics

The simplest metric for assessing the performance of a machine learning system is *accuracy*. It measures the proportion of true positives and true negatives to the total size of the sample population:

$$\frac{TruePositive + TrueNegative}{TruePositive + TrueNegative + FalsePositive + FalseNegative} \tag{2.2}$$

Two other metrics are widely used to assess performance of machine learning systems. *Precision* is the fraction of documents retrieved that are relevant:

$$\frac{TruePositive}{TruePositive + FalsePositive} \tag{2.3}$$

Recall is ratio of relevant documents retrieved to the total number of relevant documents:

$$\frac{TruePositive}{TruePositive + FalseNegative} \tag{2.4}$$

The two metrics have an inverse relationship. You can achieve 100% recall by simply classifying every sample as positive. You can achieve 100% precision by never guessing that a sample is in the class you are looking for.

Precision and recall are often combined to form a metric called the *F-Measure*. It is the harmonic mean of precision and recall:

$$2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.5}$$

The value of F-measure ranges from 0-1. An F-measure of 1 is ideal.

Area Under ROC Curve (AUC)

A receiver operating characteristics (ROC) graph displays the relation between true positives and false positives. It is often used in place of accuracy in order to compare the performance of classifiers. The vertical axis represents the true positive rate while the horizontal axis the false positive rate. Each rate can range from 0 to 1. The line y = x represents a classifier which randomly guesses a class for a given sample. When using it to analyze multinomial classes false positives represent a prediction of any other class than the correct class.

ROC curves are two-dimensional representations of classifier performance. In order to compare classifiers these curves are often reduced to a single scalar value. This is often achieved through calculating the area under the ROC curve (AUC) [6]. All points along the x = y curve have an AUC value of 0.5. This means that a classifier with an ROC value above 0.5 performs better than random chance.

[6] states that: "The AUC has an important statistical property: the AUC of a classifier is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance." Bradley's work in [7] suggests that AUC appears to be "one of the best ways to evaluate a classifier's performance on a data set when a 'single number' evaluation is required..."

2.2 Stylistics

Karlgren posits that "texts are much more than what they are about" and "[s]tyle is the difference between two ways of saying the same thing" [8]. The source of this variation comes from the linguistic choices authors make. These choices are constrained in multiple ways. Karlgren groups these variation-generating constraints into three categories. These categories are differentiated by the level of constraint they place on the author and the source of the constraint.

1. Rule (from the language): syntax and morphology

2. Convention (from the genre): lexical patterns, patterns of argumentation, and tropes

3. Free (from the author): repetition, organization, elaboration

Karlgren also provides a definition of a genre. "For most purposes, genres can be understood as groupings of documents that are a) stylistically consistent and b) intuitive to accomplished readers of the communication channel in question" [8]. In this work we suggest that linguistic genres can be linked to musical genres with the later influencing the former.

2.3 Music Genres

The words style and genre are often used interchangeably. Though some argue this is incorrect [9], we generally treat them as the same thing. Since we aim to show a link between writing style and musical genre it is necessary for us to define what a musical genre is. Meyer says that style "is a replication of patterning, whether in human behavior or in the artifacts produced by human behavior, that results from a series of choices made within some set of constraints" [10]. Fabbri and Chambers state: "A musical genre is a set of musical events, real or possible, whose course is regulated by a definite arrangement of socially accepted rules" [11]. The emphasis is my own. These definitions share a common link to our definition of writing style. Writing style and musical genres are both, in part, the products of rules and constraints. Linguistic rules may constrain things like word choice and syntax whereas musical rules may constrain things like lyrics and instrumentation. Fortunately for us SoundCloud categorizes music into several genres.

2.4 Communication Accommodation Theory

Communication Accommodation Theory (CAT) describes accommodation as "a process concerned with how we can both reduce and magnify communicative differences in inter-

action." [12] It is largely attributed to Howard Giles. It consists of two accommodative strategies of *convergence* and *divergence*.

> "*Convergence* is defined as a strategy through which individuals adapt their communicative behavior in such a way as to become more similar to their interlocutor's behavior. Conversely, the strategy of *divergence* leads to an accentuation of differences between self and other. A strategy similar to divergence is *maintenance*, in which a person persists in his or her original style, regardless of the communication behavior of the interlocutor. Central to the theory is the idea that speakers adjust (or accommodate) their speech styles in order to create and maintain positive personal and social identities." [13]

We use this theory as the basis of our motivation to search for stylistic differences and similarities between SoundCloud users.

2.5 SoundCloud Corpus

SoundCloud is an online audio sharing service headquartered in Berlin, Germany. As of July 2013, the site had 40 million registered users [14]. SoundCloud.com has an Alexa Global Rank of 171 and is ranked 145 in the United States as of May 20th, 2014 [15]. It allows users to upload audio for other users to: listen to, share, and discuss music.

SoundCloud users have the ability to comment on songs they listen to. These user comments served as the source for the corpus used in this research. SoundCloud lets users "Explore" many types of audio. We focus on the many genres of music SoundCloud hosts, including: Classical, Country, Dubstep, Electronic, Hip Hop, House, Jazz, Metal, Pop, R&B, Reggae, Rock, Techno, and World. Our corpus was compiled from user comments on songs from a subset of these genres.

The comments in our corpus were retrieved using SoundCloud's Python API wrapper documented in [16] and [17]. Each comment made on SoundCloud.com is tagged with several properties [18]:

- ID: a unique integer ID
- URI: API resource URL
- created_at: timestamp of creation
- body: HTML comment body
- timestamp: associated timestamp in milliseconds
- user_id: unique user id of the owner
- user: a group of characteristics representing the commenter
- track_id: the unique id of the audio file the user commented on

Our corpus consists of body, user_id pairs. We keep these pairs organized by their corresponding song and musical genre.

2.6 Tools

2.6.1 Waikato Environment for Knowledge Analysis

The Waikato Environment for Knowledge Analysis (Weka) is a widely used set of tools used for machine learning tasks [19]. Weka provides a GUI interface and a Java API. The GUI tool is called Weka workbench. It "includes algorithms for regression, classification, clustering, association rule mining and attribute selection". Weka also comes with implementations of several learning algorithms. We make use of it's Java API and four supervised classifiers for our experiments: Naïve Bayes, J48, IB1, and SVM. We store our training and test sets in Weka's file format called the Attribute Relation File Format (ARFF). ARFF Files contain a list of feature vectors and the corresponding description of the feature vectors included. This format allows us to extract features from comments once and use them to run multiple experiments.

2.6.2 Natural Language Toolkit

The Natural Language Toolkit (NLTK) is Python based platform for working with human language data [20]. It provides modules for:Accessing corpora, string processing, collocation discovery, part-of-speech tagging, classification, chunking, parsing, etc. We make use of its part-of-speech module. During our initial prototyping we also made use of its Naïve Bayes classifier implementation.

Chapter 3

Related Work

3.1 No Country for Old Members: User Lifecycle and Linguistic Change in Online Communities

The authors of [21] explore the linguistic change of users as they join and leave online communities. Similar to our research they target a community with well-defined, tight-knit subcultures. In this case the authors perform linguistic analysis on users of two large online communities focused on beer: BeerAdvocate and RateBeer. Their work explores "the complex interplay between community-level and individual-level linguistic change." In order to compare individuals' language to that of the community they used a series of "snapshot language models." These snapshots were created for every month in their dataset starting in 2001 and ending in 2011. They use word bigram language models with Katz back-off smoothing. Individual posts are compared to the model of the month in which they were posted. The authors calculate the post's cross-entropy with respect to the appropriate model. Posts with higher cross-entropy values are seen as deviating from the community's linguistic state.

The authors are able to leverage their technique to predict how long a user would remain active in a community by analyzing his or her first few posts. In contrast to the work in [21] we do not analyze the language of individual users nor do we treat our dataset as a singular community. Instead we examine one online community that harbors several differing subcommunities. In our case, we view a community as the users that comment on a specific

genre of music. We then explore the similarities and differences between these genre-defined communities. We also use different feature sets than [21] due to the nature of our dataset. The posts in our dataset are generally shorter and lack the structure found in their dataset.

3.2 Distinguishing Venues by Writing Styles

The authors of [22] hypothesize that venues for research paper publication are distinguishable by their writing styles. They used three standard WEKA classifiers: SVM, Naive Bayes, and Random-Forest. Using these classifiers they examined several features which they grouped into three types: lexical, syntactic, and structural. They tested their approach by randomly choosing K venues where K was one of: 2, 5, 10, 30, 50, 100, and 150. Their classification approach was able to beat random class selection for any number of venues.

This work is similar to ours in that it shows that stylometric features can be used to distinguish documents which discuss a common topic. Similar to [21] their dataset contains longer, more-structured documents than the documents in our dataset. This work helped shape our experimental setup: varying the classifier used and the number of classes.

3.3 Twitter Part-Of-Speech Tagging for All: Overcoming Sparse and Noisy Data

The authors of [23] worked to improve the quality of part-of-speech taggers when tagging tweets. SoundCloud comments and Twitter posts present similar challenges to using established NLP techniques due to their sparse and noisy nature. They use an identical scheme for representing replies: "@" followed by a username. Also, they often contain URLs and hashtags.

The authors implemented several improvements to learning-based classifiers. They were able to train their classifiers with Part-of-Speech tagged tweet datasets. Even with this datasets they often incorrectly labeled tokens. The tokens most often labeled incorrectly were: slang, jargon, misspellings, genre-related phrases, emoticons, and unambiguous named

entities. They used a lookup list in order to translate many unknown words to more common forms (e.g. luv to love). For words not in the list they applied the use of "rare word features" such as word shape and word length. The use of fixed-word lists alone repaired approximately 80% of unknown tokens.

Further improvements to PoS-tagging came from a bootstrapping approach. They used their labeled training data to automatically tag tweets from the streaming 10% of global tweets that the Twitter API provides. If their multiple taggers agreed upon labels for a tweet's tokens then they included the tweet in the set of tweets that would a be used to train the taggers in future runs. This approach improved their accuracy to 90.54% for tokens and 28.81% for sentences. This was after the taggers were trained with 1.5 million tokens. Their data suggests the taggers could be improved with more training tokens.

Though we do not use their PoS tagging scheme, we use their work to decide which tokens to substitute.

Chapter 4

Methodology

4.1 Text Processing

User comments are processed in several steps in order to put them in the proper format for classification.

1. Read comments from disk.
2. Tokenize each comment.
3. Part-of-Speech Tagging (optional).
4. Substitute certain tokens
5. Stem tokens that weren't substituted.

4.1.1 Tokenization and Token Substitution

Tokenizers made for literature and prose do not work well on our corpus. This is due to token types unique to many forms of text on the web. In order to break our text into useful tokens we use a regular expression based tokenizer developed by Christopher Potts [24] and improved upon by Jganadg Gopinadhan [25]. We look for the following types of tokens:

- Emoticons. I.e. ":)"
- Hashtags. I.e. "#music"

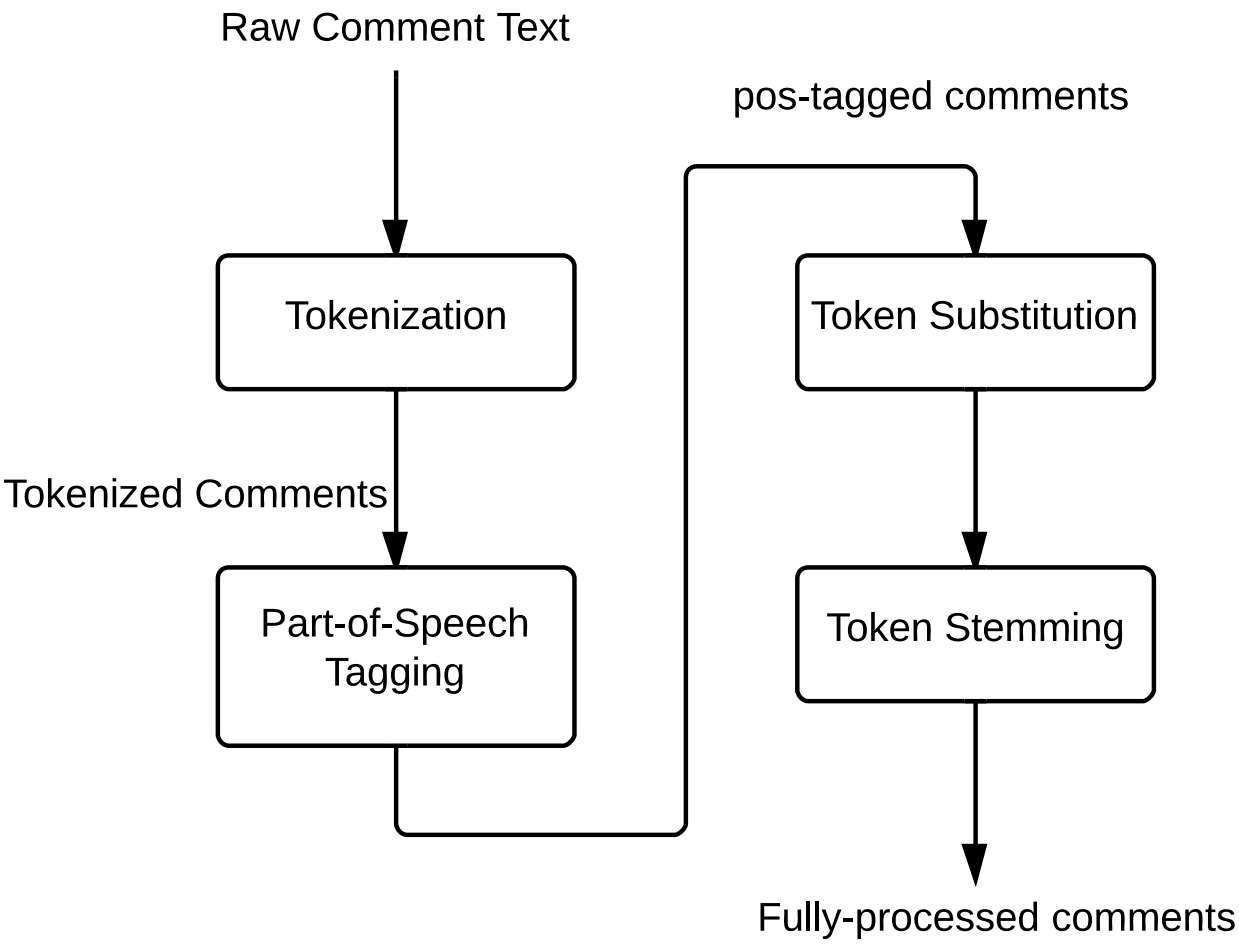

Figure 4.1: Text Processing Pipeline

- Replies to other users. I.e. "@username"
- URLs
- Words with apostrophes or dashes.
- Numbers
- Words without apostrophes or dashes.
- Ellipsis.
- Everything else that isn't whitespace.

We replace many tokens with more generic tokens. Emoticons are replaced with [EMOTICON], hashtags are replaced with [HASHTAG], replies are replaced with [REPLY], and URLs are replace with [URL]. We also replace musical terms with [MUSICAL_TERM]. This token substitution is used to collapse several tokens to improve performance for unigram and bigram feature sets.

4.1.2 Part-of-Speech Tagging

Following tokenization and prior to stemming we use the Natural Language Toolkit's (NLTK) [20] part-of-speech tagger based on the Penn Treebank [26] tag set. We keep track of part-of-speech unigrams and bigrams for use in feature extraction.

4.1.3 Stemming

All tokens that haven't been replaced with more generic tokens are stemmed to collapse them onto a smaller set of tokens. We make use of the Porter Stemming Algorithm [27]. The Porter stemmer "removes common morphological and inflexional endings from words in English." Similar to token substitution this process improves unigram and bigram feature sets.

4.1.4 Term Frequency - Inverse Genre Frequency

In order to select the most useful token unigrams and bigrams for classification we apply a well-known technique called Term Frequency - Inverse Document Frequency (TFIDF). The process of TFIDF generates a score per token in a document. This score is the token's frequency in the document divided the number of documents in the corpus that contain it. In our case we treat each genre as a single document. This approach gives us a ranking of each token's uniqueness to the genre it is found in. We call this approach Term Frequency - Inverse Genre Frequency (TFIGF). In our case the highest TFIGF scores represent tokens which show up many times in a single genre and/or show up in few genres. Low TFIGF

scores represent tokens that show up in many of the genres and/or are used less frequently.

We choose the tokens with the highest TFIGF scores in each genre to use as features. We also complete the TFIGF process for token bigrams to select the most relevant bigrams for use as features in classification. In order to cap the dimensionality of our feature set we only use the top 100 token unigrams and bigrams from each genre as features. This number was chosen through experimentation. It outperformed using the top 50 and top 150 unigrams and bigrams.

4.1.5 Feature Extraction and Feature Families

Machine learning algorithms require a feature vector which represents value of each feature in a document. In order to create this vector we "extract" the values for each document. In this section we describe the features we use and the process of extracting them.

Token N-Grams

We check for the presence of each of the token unigrams selected by the TFIGF process. The value of this feature is "true" if the comment contains the token or "false" if the token is not found in the comment.

Like the unigrams we check for the simple prescence of each of the bigrams selected by the TFIGF process. The value of this feature is "true" if the comment contains the token bigram or "false" if the token bigram is not found in the comment.

Counts

We check for the presence of repeated character strings. This feature represents whether or not a comment contains a string of 3 or more of the same characters in a row. A comment containing "cool" would receive the value "false" for this feature. A comment containing "coool", "coooool", etc would receive a value of "true" for this feature. This feature is extracted using the original comment (prior to tokenization). One version of the feature

represents the presence or absence of repeated letters. The other version of this feature represents the presence or absence of punctuation.

We use the length of the comment in characters and the number of tokens it contains as features. We also use the ratio of unique tokens to total number of tokens as a feature

Dictionaries

The dictionary based features consist of the presence or absence of tokens from three separate dictionaries. The first dictionary is the simplest. It consists of the names of all genres included in the experiment. We also check for the presence or absence of musical terms taken from a list of terms on Wikipedia [28]. Finally we check for the presence or absence of emoticons from a list found on Wikipedia [29].

Part-of-Speech N-Grams

We check for the presence or absence of each of the part-of-speech tags found in the Penn Treebank [26]. We also check for the presence or absence of every permutation of Penn Treebank part-of-speech tags.

Chapter 5

Experimental Setup

We ran a total of 180 classification experiments. Our initial experiments were used to determine the best feature set and classifier. We ran these experiments with 8 different genres: Classical, Hip Hop, Jazz, Country, Metal, Folk, World. SoundCloud provides an interface for browsing audio by genre. This interface provides 37 genres to choose from. Of these genres 11 are non-music audio and at least 13 can be considered sub-genres of Electronic music. We selected our 8 from the remaining 13 genres for their relative distinctness from each other. We held the number of training and test comments constant for each genre.

We grouped our features into four different sets: counts, dictionaries, part-of-speech n-grams, and token n-grams. Counts consists of: the number of tokens, the length of a comment in characters, the ratio of unique tokens to total tokens, and the presence of repeated characters. Dictionaries consists of: the presence of or absence of the genre names themselves, the presence or absence of musical terms, and the presence or absence of emoticons. Part-of-speech n-grams consists of the presence or absence of each of the parts-of-speech in the Penn Treebank as well as the presence or absence of each combination of two parts-of-speech from the Penn Treebank. Token n-grams consists of the top scoring unigrams and bigrams selected from the TF-IGF process. We also experimented with using all of these features together.

Once our feature sets were determined we experimented with several data sets consisting of 2 or 4 genres. We randomly selected 20 combinations of 2 genres and 20 combinations of 4 genres for these runs. As we will show in the Results chapter our best feature set consisted

of token unigrams and bigrams.

Our experiments were run on a desktop machine with: an Intel i5-2500k quad-core processor clocked at 3.3 GHz and 12 GB of DDR3 RAM. It used Ubuntu 14.04 LTS as the operating system. No experiment was limited by memory though many were bottlenecked by the CPU.

Chapter 6

Results

6.1 Feature Set and Classifier Combinations

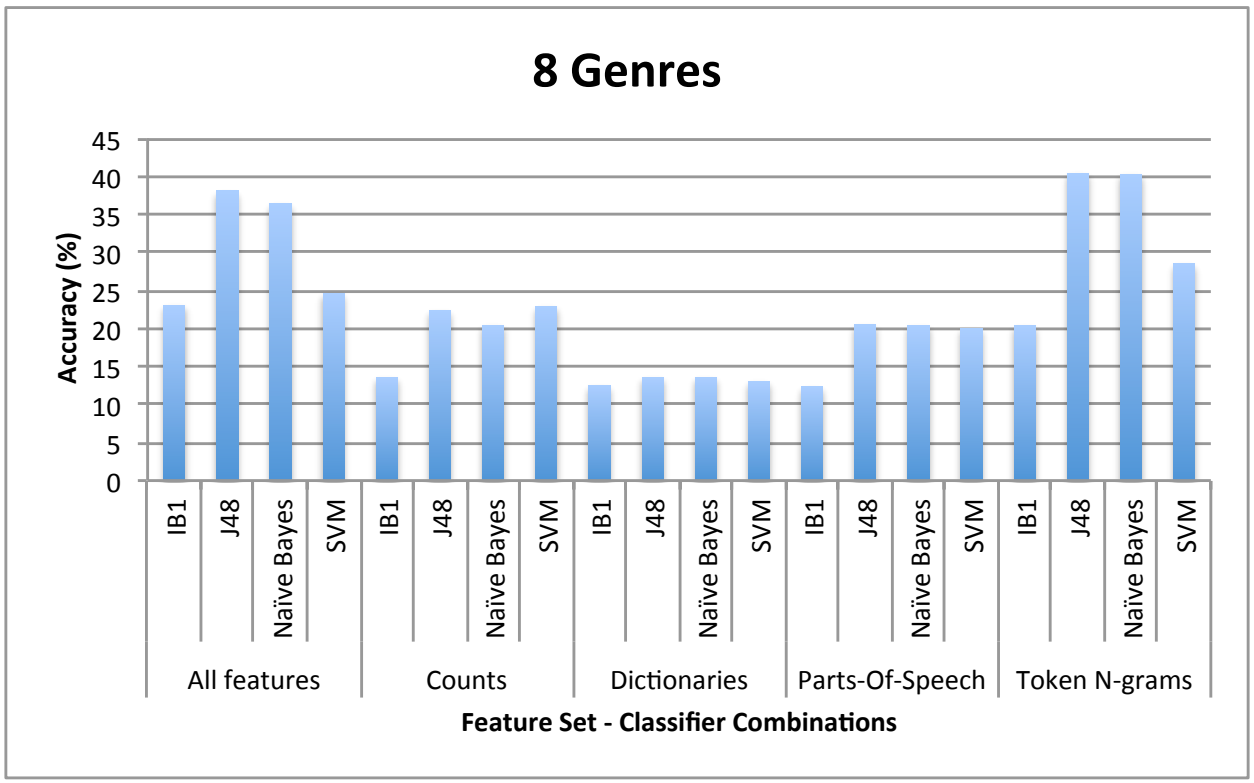

Figure 6.1: Results for 8 Genre Experiments

For all feature sets the J48 and Naïve Bayes classifiers performed the best. Neither classifier was clearly the best performer. Though we were generally unconcerned with the run time of experiments it is interesting to note that J48 experiments took up to 24 hours to complete while Naïve Bayes experiments took 1 to 2 minutes to complete. In generally Naïve

Bayes experiments were the quickest to run. The IB1 classifier clearly performed the worst of all classifiers. This was expected since the IB1 classifier simply tries to find the single training comment that a test comment is most similar to and uses the training comment's genre as its prediction.

Our experiments show that the feature set consisting of token unigrams and bigrams performed the best. It was followed closely by a feature set consisting of all of our features. The hand-made dictionary feature set performed around random chance levels while all other feature sets outperformed random chance.

6.2 Varying the Number of Genres

Our experiments with 2 and 4 genre sets confirm that the J48 and Naïve Bayes classifiers demonstrate the best performance. On average we achieve a 1.5x improvement over chance for 2 genre classification experiments: 75% vs 50%. For 4 genre experiments we achieve 2x improvement over chance: 50% vs 25%. These results suggest that we could reliably detect the genre of a song based on the comments made about it

6.2.1 Statistical Significance Tests

We compare our results against a classifier which randomly selects a class label and against a Naïve Bayes classifier trained with unmodified unigram tokens.

2 Genres

Our system achieved a mean accuracy of 73.2% with a standard deviation of 8.04 for 28 runs. To compare our results to a random choice classifier we generated a data set containing 960 test samples evenly divided between two classes. Over 100 runs this classifier achieved a mean accuracy of 50.01% with standard deviation of 1.53. Our null hypothesis was that our classification scheme would achieve the same accuracy as the random classifier. Our alternative hypothesis was that our classifier would achieve a higher accuracy than the

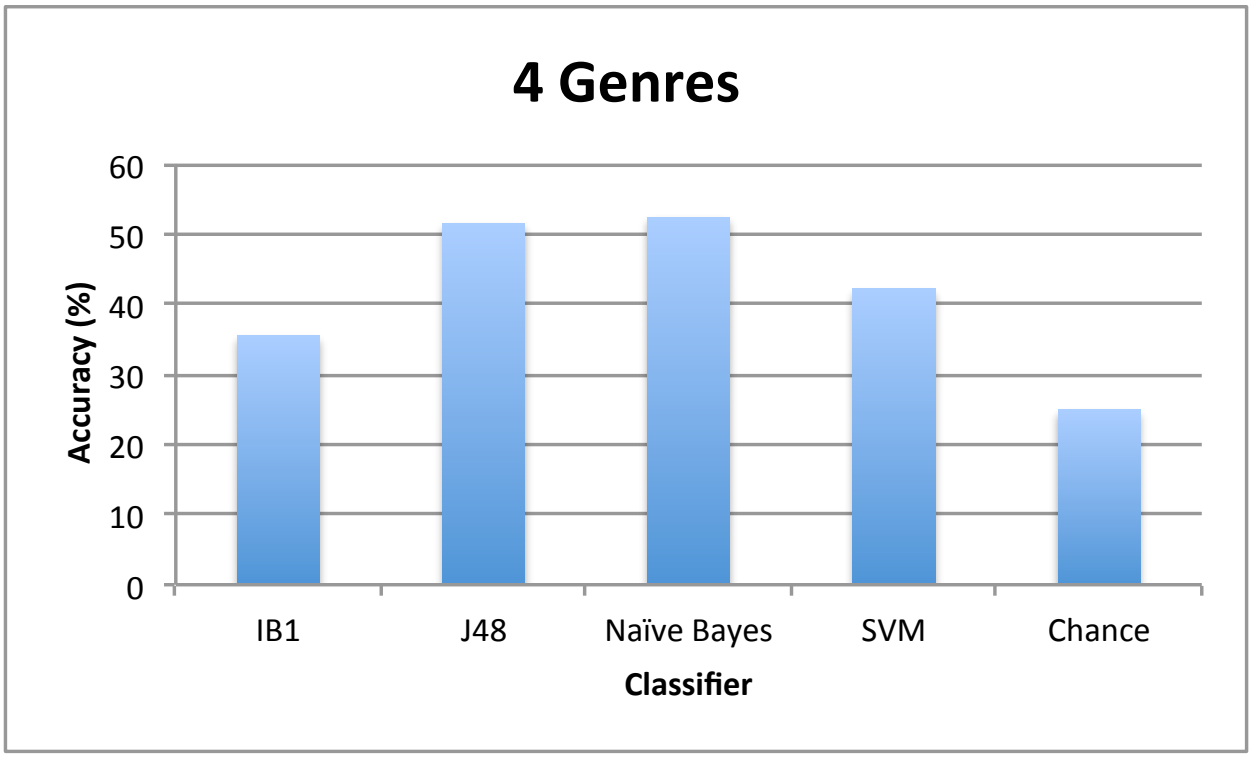

Figure 6.2: Results for 4 Genre Experiments

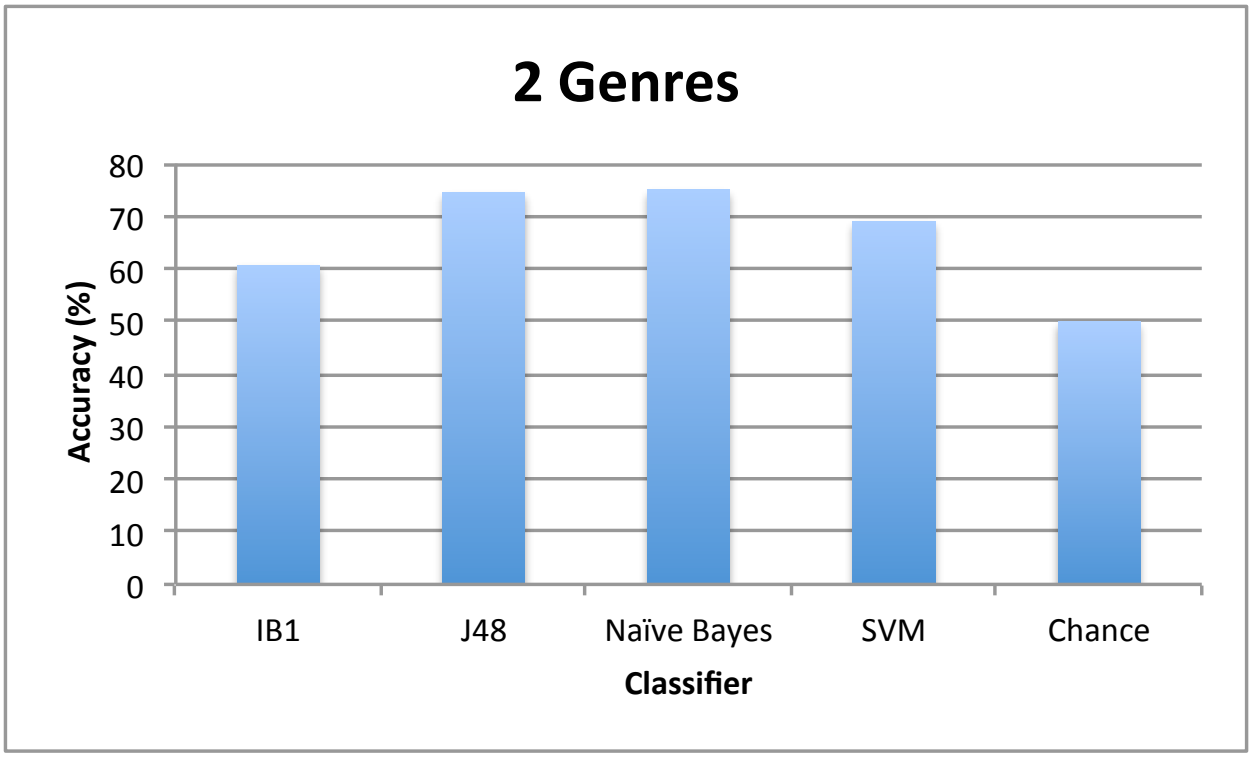

Figure 6.3: Results for 2 Genre Experiments

random classifer. We test for statistical significance using a Z-test. With these values we achieve a z-score of 15.35. Our p-value is far less than 0.01. This leads us to reject the null hypothesis and accept our alternative hypothesis that our classifier performs better than a random choice classifier.

We ran 30 experiments with a Naïve Bayes classifier trained on the 250 most frequent unigram tokens found in the training set. This approach achieved a mean accuracy of 69.4% with a standard deviation of 7.97. This results in a z-score of 2.51 and p-value of 0.006. This is also far below the 0.05 threshold for statistical significance showing that our classifier outperforms a baseline unigram approach.

4 Genres

For the 4 genre case we test our classifier against the same random classifier and Naïve Bayes classifier trained on token unigrams as before. Our system achieved a mean accuracy of 51.40% with a standard deviation of 5.67 over 30 runs. The random choice classifier achieved a mean accuracy of 24.97% with a standard deviation of 1.08. This results in a z-score of 25.65 and a p-value far below the .05 statistical significance threshold.

The Naïve Bayes classifier trained on unigrams achieved a mean accuracy of 46.51% and a standard deviation of 5.24. This results in a z-score of 2.51 and p-value of 0.0048. This is under the threshold of 0.05 for statistical significance.

6.3 Per-Genre AUC Results

The data in table 6.1 shows how well our classification system can identify each individual genre. According to these results the "World" genre is most easily identified relative to other genres. This seems counterintuitive because the "World" genre encompasses a wide range of genres: usually music of non-western origin. It is also strange because the poorest performance genre, "Electronic," also represents a wide range of sub-genres: essentially any music that is made up of "electronically produced or modified sounds" [30]. For most

Genre	2 Genre Experiment	4 Genre Experiment	8 Genre Experiment
Hip Hop	0.86	0.81	0.81
Classical	0.82	0.78	0.77
Country	0.81	0.76	0.72
Jazz	0.79	0.72	0.69
Metal	0.80	0.76	0.76
World	0.93	0.91	0.93
Electronic	0.75	0.73	0.75
Folk	0.80	0.77	0.76

Table 6.1: Per Genre ROC results using best feature set and classifier.

genres performance decreases slightly as the total number of genres increases. However the difference in performance is relatively small.

Chapter 7

Recommendations for Future Work

There are several potential improvements, applications, and sources of further analysis for this work. We ignored the temporal variability of user writing style. The technique in [21] could be applied to a SoundCloud corpus collected over an extended period of time in order to see the change in writing style of users on a per-genre level.

We previously mentioned that SoundCloud contains nearly 15 sub-genres of "Electronic" music. Our technique could be applied at the sub-genre level. This would allow us to determine how much variation exists within genres. Our system could be trained using musicians as class labels. This could be used to determine if fans of artists accomadate each others' langauge.

Chapter 8

Contributions

We now present a summary of the contributions of this work.

- A corpus of user comments from SoundCloud
- Term Frequency-Inverse Genre Frequency: a modification of the tf-idf statistic
- Analysis of per-genre prediction performance.
- A system for predicting the genre of online user comments

Chapter 9

Conclusion

Comments on SoundCloud show similarity to comments found on other social media sites such as Twitter. They are often short and contain: frequent misspellings, emoticons, hashtags, URLS, etc. This allows us to apply techniques used on Tweet-based corpora to our own corpus of SoundCloud comments. We examined the efficacy of several different feature sets. In the end we found that using a feature set consisting of token unigrams and bigrams provided the highest performance as measured by accuracy. Our classification system outperforms a random chance and a baseline Naïve Bayes classifier trained on unigrams within a margin of statistical significance. We were surprised to find that the combination of all of our features did not perform better than the feature set of unigrams and bigrams. We thought that the combination of all features would improve performance because each individual feature set was able to provide improvement over random chance (though likely not statistically significant). We hypothesize that this could be attributed to the Curse of Dimensionality. In the context of machine learning this curse states that adding more features (dimensions) without adding more training samples can actually hurt performance.

Our results suggest that users of SoundCloud show *some* level of genre-dependent writing style. Per-genre AUC analysis suggests that some genres are easier to distinguish than others. In our corpus the "World" genre was the most easily distinguished from other genres.

BIBLIOGRAPHY

[1] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. The MIT Press, 2012.

[2] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[3] C.-C. C. Chih-Wei Hsu and C.-J. Lin, *A Practical Guide to Support Vector Classification*, National Taiwan University, Taipei: Department of Computer Science, April 2010.

[4] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, May 2011. [Online]. Available: http://doi.acm.org/10.1145/1961189.1961199

[5] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Mach. Learn.*, vol. 6, no. 1, pp. 37–66, Jan. 1991. [Online]. Available: http://dx.doi.org/10.1023/A:1022689900470

[6] T. Fawcett, "An introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, December 2005.

[7] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, pp. 1145–1159, 1997.

[8] J. Karlgren, "The wheres and whyfores for studying textual genre computationally."

[9] A. F. Moore, "Categorical conventions in music discourse: Style and genre," *Music & Letters*, vol. 82, no. 3, pp. pp. 432–442, 2001. [Online]. Available: http://www.jstor.org/stable/3526163

[10] L. Meyer, *Style and Music: Theory, History, and Ideology*, ser. Studies in the criticism and theory of music. University of Chicago Press, 1989. [Online]. Available: http://books.google.com/books?id=hPksngEACAAJ

[11] F. Fabbri and I. Chambers, "What kind of music?" *Popular Music*, vol. 2, pp. pp. 131–143, 1982. [Online]. Available: http://www.jstor.org/stable/852979

[12] L. A. Baxter and D. O. Braithwaite, *Engaging Theories in Interpersonal Communication*. SAGE Publications, Inc., 2008.

[13] W. B. Gudykunst, "Theorizing about intercultural communication," September 2004.

[14] J. Graham, "Who's listening to soundcloud? 200 million," July 2013. [Online]. Available: http://www.usatoday.com/story/tech/columnist/talkingtech/2013/07/17/whos-listening-to-soundcloud-200-million/2521363/

[15] "How popular is soundcloud.com?" May 2014. [Online]. Available: http://www.alexa.com/siteinfo/www.soundcloud.com

[16] [Online]. Available: https://github.com/soundcloud/soundcloud-python

[17] SoundCloud.com. [Online]. Available: http://developers.soundcloud.com/docs

[18] ——. [Online]. Available: http://developers.soundcloud.com/docs/api/reference#comments

[19] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1656274.1656278

[20] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. O'Reilly Media, 2009.

[21] C. Danescu-Niculescu-Mizil, R. West, D. Jurafsky, J. Leskovec, and C. Potts, "No country for old members: User lifecycle and linguistic change in online communities," in *Proceedings of the 22Nd International Conference on World Wide Web*, ser. WWW '13. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2013, pp. 307–318. [Online]. Available: http://dl.acm.org/citation.cfm?id=2488388.2488416

[22] Z. Yang and B. D. Davison, "Distinguishing venues by writing styles," in *Proceedings of the 12th ACM/IEEE-CS Joint Conference on Digital Libraries*, ser. JCDL '12. New York, NY, USA: ACM, 2012, pp. 371–372. [Online]. Available: http://doi.acm.org/10.1145/2232817.2232896

[23] L. Derczynski, A. Ritter, S. Clark, and K. Bontcheva, "Twitter part-of-speech tagging for all: Overcoming sparse and noisy data." in *RANLP*, G. Angelova, K. Bontcheva, and R. Mitkov, Eds. RANLP 2011 Organising Committee / ACL, 2013, pp. 198–206. [Online]. Available: http://dblp.uni-trier.de/db/conf/ranlp/ranlp2013.html#DerczynskiRCB13

[24] C. Potts, 2011. [Online]. Available: http://sentiment.christopherpotts.net/code-data/happyfuntokenizing.py

[25] J. Gopinadhan, June 2013. [Online]. Available: https://bitbucket.org/jaganadhg/twittertokenize/src/

[26] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of english: The penn treebank," *COMPUTATIONAL LINGUISTICS*, vol. 19, no. 2, pp. 313–330, 1993.

[27] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, July 1980.

[28] Wikipedia, April 2014. [Online]. Available: http://en.wikipedia.org/wiki/Glossary_of_jazz_and_popular_music

[29] ——, April 2014. [Online]. Available: http://en.wikipedia.org/wiki/Emoticon

[30] T. B. Holmes, *Electronic and Experimental Music: Pioneers in Technology and Composition.* Psychology Press, 2002.

Appendix A

Full Experimental Results

Genres	**IB1**	**J48**	**Naive Bayes**	**SVM**
Classical/World	69.5	84.8	85.4	79.7
Jazz/Electronic	53.6	63.8	65.2	58.8
Jazz/World	68.1	84.9	86.7	79.8
Hip Hop/World	63.9	82.0	82.2	79.2
Jazz/Metal	56.0	67.4	65.9	59.1
Country/Folk	55.7	64.2	63.8	60.4
Classical/Metal	62.0	74.9	74.0	67.5
World/Folk	69.0	84.4	85.5	78.0
Hip Hop/Folk	61.6	77.9	77.6	71.1
Classical/Country	58.8	71.1	70.1	65.6
Country/Hip Hop	59.8	75.1	76.9	68.4
Classical/Hip Hop	63.6	78.8	78.5	71.8
Jazz/Hip Hop	56.5	73.8	74.3	65.9
Country/World	59.2	86.4	86.9	80.2
Metal/World	68.1	84.2	86.0	81.0
Classical/Folk	56.4	67.1	66.2	61.0
Classical/Electronic	61.6	70.3	73.3	67.0
Metal/Electronic	54.5	63.8	64.3	61.2
Classical/Jazz	58.5	65.8	67.2	60.0
Metal/Folk	58.2	70.4	70.2	65.5

Table A.1: Accuracy (%) for Two Genre Experiments using Token Unigrams and Bigrams.

Genres	IB1	J48	Naive Bayes	SVM
Classical/World	0.667	0.847	0.854	0.794
Jazz/Electronic	0.438	0.637	0.641	0.543
Jazz/World	0.649	0.848	0.866	0.795
Hip Hop/World	0.588	0.82	0.821	0.789
Jazz/Metal	0.472	0.671	0.653	0.561
Country/Folk	0.471	0.627	0.632	0.591
Classical/Metal	0.568	0.748	0.734	0.65
World/Folk	0.661	0.843	0.854	0.778
Hip Hop/Folk	0.561	0.779	0.776	0.696
Classical/Country	0.52	0.706	0.692	0.639
Country/Hip Hop	0.523	0.75	0.768	0.666
Classical/Hip Hop	0.587	0.787	0.783	0.704
Jazz/Hip Hop	0.477	0.737	0.741	0.643
Country/World	0.515	0.862	0.868	0.799
Metal/World	0.65	0.841	0.859	0.808
Classical/Folk	0.493	0.669	0.649	0.584
Classical/Electronic	0.561	0.702	0.726	0.643
Metal/Electronic	0.451	0.626	0.635	0.588
Classical/Jazz	0.508	0.652	0.661	0.567
Metal/Folk	0.518	0.703	0.701	0.641

Table A.2: F-Measure for Two Genre Experiments using Token Unigrams and Bigrams.

Genres	IB1	J48	Naive Bayes	SVM
Country/Metal/World/Folk	34.8	56.3	57.4	47.4
Classical/Jazz/Metal/World	40.6	56.3	58.5	46.7
Jazz/Hip Hop/World/Electronic	39.6	56.6	57.7	46.0
Country/Hip Hop/World/Electronic	33.2	57.1	57.9	47.2
Metal/World/Electronic/Folk	39.0	57.8	58.0	46.0
Classical/Jazz/Hip Hop/World	41.6	58.1	60.0	49.2
Classical/Country/Jazz/Hip Hop	34.4	49.1	51.2	39.1
Classical/Jazz/Hip Hop/Folk	33.0	47.7	49.3	38.6
Jazz/Hip Hop/Electronic/Folk	33.8	47.6	47.6	36.1
Classical/Country/World/Folk	35.2	54.1	55.5	45.2
Classical/Jazz/Metal/Folk	33.3	43.7	45.0	36.2
Country/Jazz/Hip Hop/Electronic	32.7	46.9	47.0	36.9
Jazz/Metal/Hip Hop/Folk	34.4	48.2	47.5	40.1
Country/Hip Hop/World/Folk	35.4	59.3	60.6	49.6
Classical/Jazz/Metal/Electronic	33.6	44.9	46.4	36.0
Metal/Hip Hop/World/Electronic	39.8	53.0	54.3	45.6
Classical/Jazz/Metal/Hip Hop	34.7	49.3	49.9	40.5
Classical/Country/Metal/Hip Hop	36.4	51.4	51.1	43.5
Metal/Hip Hop/Electronic/Folk	33.7	48.5	48.7	38.6
Country/Jazz/Metal/Electronic	31.1	45.2	44.5	35.3

Table A.3: Accuracy (%) for Four Genre Experiments using Token Unigrams and Bigrams.

Genres	IB1	J48	Naive Bayes	SVM
Country/Metal/World/Folk	0.296	0.563	0.572	0.446
Classical/Jazz/Metal/World	0.348	0.558	0.581	0.42
Jazz/Hip Hop/World/Electronic	0.335	0.568	0.581	0.423
Country/Hip Hop/World/Electronic	0.264	0.573	0.583	0.448
Metal/World/Electronic/Folk	0.327	0.578	0.583	0.427
Classical/Jazz/Hip Hop/World	0.366	0.576	0.599	0.465
Classical/Country/Jazz/Hip Hop	0.293	0.478	0.5	0.35
Classical/Jazz/Hip Hop/Folk	0.274	0.46	0.479	0.34
Jazz/Hip Hop/Electronic/Folk	0.278	0.475	0.475	0.317
Classical/Country/World/Folk	0.298	0.54	0.556	0.448
Classical/Jazz/Metal/Folk	0.273	0.428	0.437	0.32
Country/Jazz/Hip Hop/Electronic	0.265	0.472	0.472	0.335
Jazz/Metal/Hip Hop/Folk	0.287	0.468	0.474	0.373
Country/Hip Hop/World/Folk	0.3	0.589	0.604	0.467
Classical/Jazz/Metal/Electronic	0.272	0.444	0.461	0.307
Metal/Hip Hop/World/Electronic	0.34	0.529	0.547	0.429
Classical/Jazz/Metal/Hip Hop	0.295	0.483	0.499	0.376
Classical/Country/Metal/Hip Hop	0.32	0.502	0.515	0.426
Metal/Hip Hop/Electronic/Folk	0.279	0.486	0.494	0.367
Country/Jazz/Metal/Electronic	0.242	0.448	0.439	0.309

Table A.4: F-Measure for Four Genre Experiments using Token Unigrams and Bigrams.

Feature Set	IB1	J48	Naive Bayes	SVM
Word	20.3	40.4	40.3	28.5
Counts	13.6	22.4	20.4	22.9
Dictionaries	12.6	13.6	13.6	13.1
POS	12.4	20.5	20.4	20.0
All Features	23.1	38.1	36.4	24.6

Table A.5: Accuracy (%) for each Feature Set with Eight Genres

Feature Set	IB1	J48	Naive Bayes	SVM
Word	0.178	0.402	0.401	0.247
Counts	0.062	0.209	0.14	0.212
Dictionaries	0.029	0.05	0.05	0.039
POS	0.035	0.153	0.137	0.149
AllFeatures	0.226	0.374	0.358	0.22

Table A.6: F-Measure for each Feature Set with Eight Genres